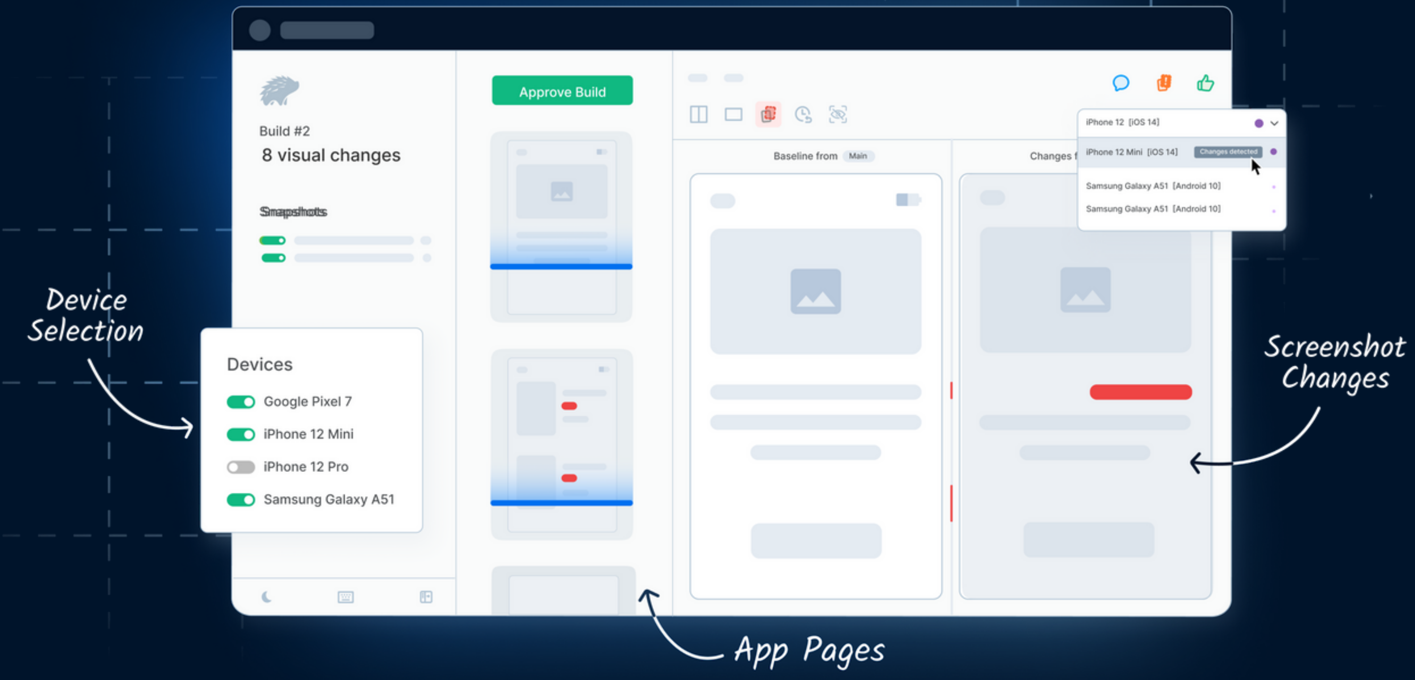
BrowserStack

Build and Deploy Flawless Frontends with Automated Visual Testing



TRUSTED BY GLOBAL ENTERPRISES

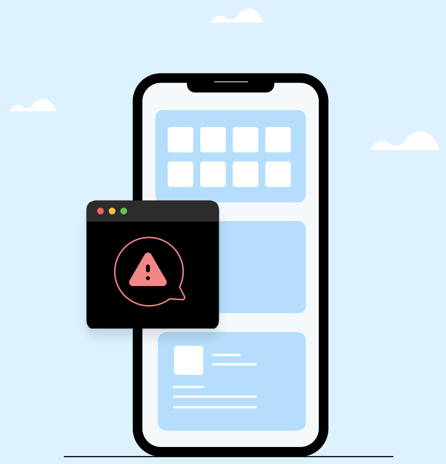
mastercard · Microsoft · The Washington Post · RBS · CISCO · HSBC · Discovery

Build and Deploy Flawless Frontends with Automated Visual Testing

It takes only 50 milliseconds (that's 0.05 seconds) to decide if users love your website and stay or give it a pass and leave.

More people use smartphones and tablets than desktops, making mobile usage dominant. To succeed, apps and websites must offer top-notch experiences on diverse mobile devices.

The stats don't lie.



- 61% of users are unlikely to return to a site on mobile if they had trouble accessing it and 40% visit a competitor's site instead.
  Source: **McKinsey**

- 53% of mobile website visits are abandoned if pages take longer than 3 seconds to load.
  Source: **Google**

- 58% of all multi-device purchases use mobile to close the sale.
  Source: **MerchantSavvy**

Inconsistencies across browsers, devices, and screen resolutions create a fragmented user experience with visual defects, layout issues, and broken elements. It may also lead to slow loading times and increased user bounce rates. This frustration diminishes engagement and conversion opportunities.

So what do teams do to address these visual issues? They fall back to what they know: throwing more humans, engineering time, and inadequate tooling at the problem.

They face three main challenges:

1. Manual testing is time-consuming and inconsistent

Manual testing, whether conducted internally or outsourced, poses challenges due to potential human errors.

Testers may overlook issues or misinterpret visual cues, leading to inaccurate results, and impacting overall test coverage and product reliability.

It can also be time-consuming, especially for complex apps or websites, requiring manual checks for each page or screen.

Another drawback is the inconsistency in evaluation, with different testers, yielding varying results.

RCA (root cause analysis) also takes a hit, delaying the testing process and resolution of critical defects. Lastly, the cost of manual Visual Testing can be substantial, particularly with a large number of testers required for comprehensive testing.

2. Functional tests are not meant to assess visual aspects

Functional testing, though incredibly important, is simply not designed to test applications from a visual standpoint. Instead of looking at the visuals, they only test the abstractions underneath. Trying to bolt on assertions of visual aspects doesn't work and creates fragile and unmaintainable test suites.

Automated Visual Testing is similar to functional testing in that it's designed to be an automated process that runs alongside code reviews. Unlike functional testing, however, visual tests don't pass or fail. Automated Visual Testing simply detects visual changes and provides a review process to determine whether or not the changes are correct.

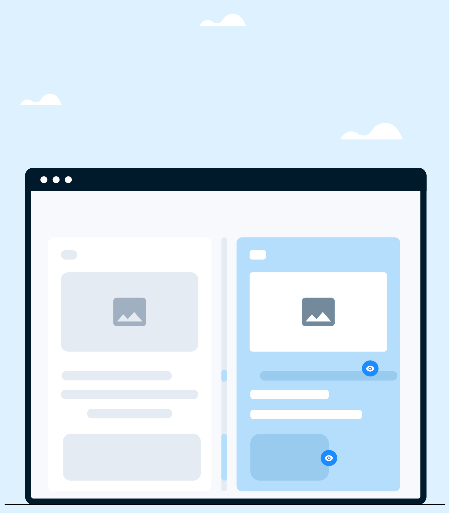
3. Open-source libraries are unmaintained and narrow

Open-source libraries have cropped up over the years, providing simple snapshotting and image diffing capabilities, but lack the infrastructure and workflow necessary to enable continuous visual coverage. They can be difficult to integrate with your existing testing infrastructure, so automating the process of Visual Testing can be challenging. You end up doing manual testing, which can be time-consuming and error-prone.

Open-source libraries are often unmaintained. They may not be updated with new features or bug fixes quickly. They are also often narrow, supporting a limited number of browsers, devices, or screen resolutions.

The most pressing challenges that keep frontend engineers on their toes

In a Visual Testing survey conducted, we asked frontend engineers what are their most difficult challenges at work. These were the responses:



- **Maintaining Cross-browser compatibility:** 38% of surveyed want to ensure websites or apps work smoothly across multiple browsers.
- **Manual UI consistency checks:** 30% of respondents said maintaining a consistent user interface across different screens, resolutions, and devices is a challenge.
- **Writing and managing tests:** 16% voted that designing comprehensive test suites, crafting solid test cases, and keeping track of test coverage is a challenge.
- **Refactoring challenges and fear of breaking:** 16% want to find safe and effective ways to refactor code, ensuring that existing functionality remains intact while bringing in much-needed improvements.

Uncover Visual Defects with Automated Visual Testing

As technology and tooling have evolved, automated visual testing has emerged as the most scalable and holistic approach to testing visual elements of frontends. From day-to-day operations to specific use cases, automated visual testing helps teams save time and get more confidence in their UI.

In our survey, we asked which benefits of automated visual testing are the most important. These are the top four which were highlighted:

1. Ensuring UI/UX Consistency across Devices and Screens

Automated Visual Testing provides continuous visual feedback regardless of the context of the change or the “correctness” of the change. In situations where your UI shouldn’t change (like deleting CSS, refactoring CSS, or upgrading dependencies), Automated Visual Testing gives you confidence that your entire UI has remained stable. That built-in coverage not only extends to the breadth of your UI—from your most critical flows down to your 404 page—but also to the combinations of those pages across browsers and screen sizes.

2. Navigating the Blind Spots in QA with Visual Regression Testing

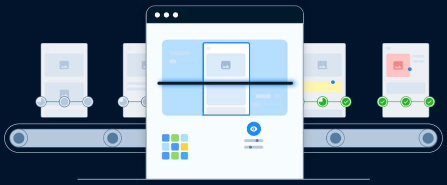
Visual regression tests are vital for preventing visual bugs from slipping into production and disrupting user experiences. While functional and manual testing catch bugs, they often don’t discover bugs, especially visual discrepancies.

Let’s go back to our survey. We asked people how often they perform regression testing on their native apps and websites.

Frequency of Visual Regression Testing	Website	Apps
Daily	30%	26%
Twice a Week	16%	21%
Once a Week	27%	23%
Less Frequently	27%	30%

A striking trend that emerges from the survey is the substantial percentage of respondents who engage in regression testing less frequently—30% for native apps and 27% for websites. This is concerning, as it highlights a potential blind spot in the QA process.

When questioned on the number of visual regression tests run, an alarming 32% admitted to not running any visual regression tests at all, which can potentially result in a deterioration of user experience and functionality over time.



How many visual regression tests do you run?

1 - 500: 37%	> 1000: 13%
500 - 1000: 18%	**We don't run them: 32%**

Overall the survey stresses the adage of "test early, test often". Infrequent testing creates an environment where bugs and defects can accumulate unnoticed, only to surface later when they're more challenging and costly to address.

Automated Visual Testing fills the gap by providing a systematic approach to ensuring consistent and frequent regression testing. By comparing screenshots across different devices and resolutions, QA teams can meticulously compare UI elements and screens against established baselines, detecting even the slightest deviations that could indicate bugs or unintended changes.

3. Increased Productivity and Scalability for QA Teams

Automated Visual Testing scales with a level of precision not feasible with the human eye, at a faster rate than the brain can work, and at a fraction of the cost. It has virtually no limitations to scale and has a low incremental cost that doesn't increase exponentially with complexity.

The time-saving and confidence-boosting benefits of Automated Visual Testing also extend to teammates other than developers. For designers verifying the implementation of designs, product managers staying in the loop, or product marketers grabbing updated UI screenshots, Automated Visual Testing provides immense value to cross-functional product teams.

4. Getting Complete Code Refactoring Confidence

When making code changes, it's often a fear of the unknown that causes the most stress—whether it's fear of breaking something or uncertainty about the full scope of design implementation. That risk, and the inability to mitigate it, is why teams spend time and resources manually looking for visual bugs. It's also why teams end up trying to write functional tests to prevent visual regressions.

Visual Testing automates that work, empowering you to merge and deploy with full confidence that your app will look exactly as it should across browsers and screen sizes.

A Shift in Mindset: Visual - First Automation is the Way Forward

In the journey towards comprehensive test automation, teams often prioritize functional test automation before automated visual testing. A more effective strategy calls for reversing this sequence and emphasizing automated visual tests as a primary focus during the automation ramp-up phase.

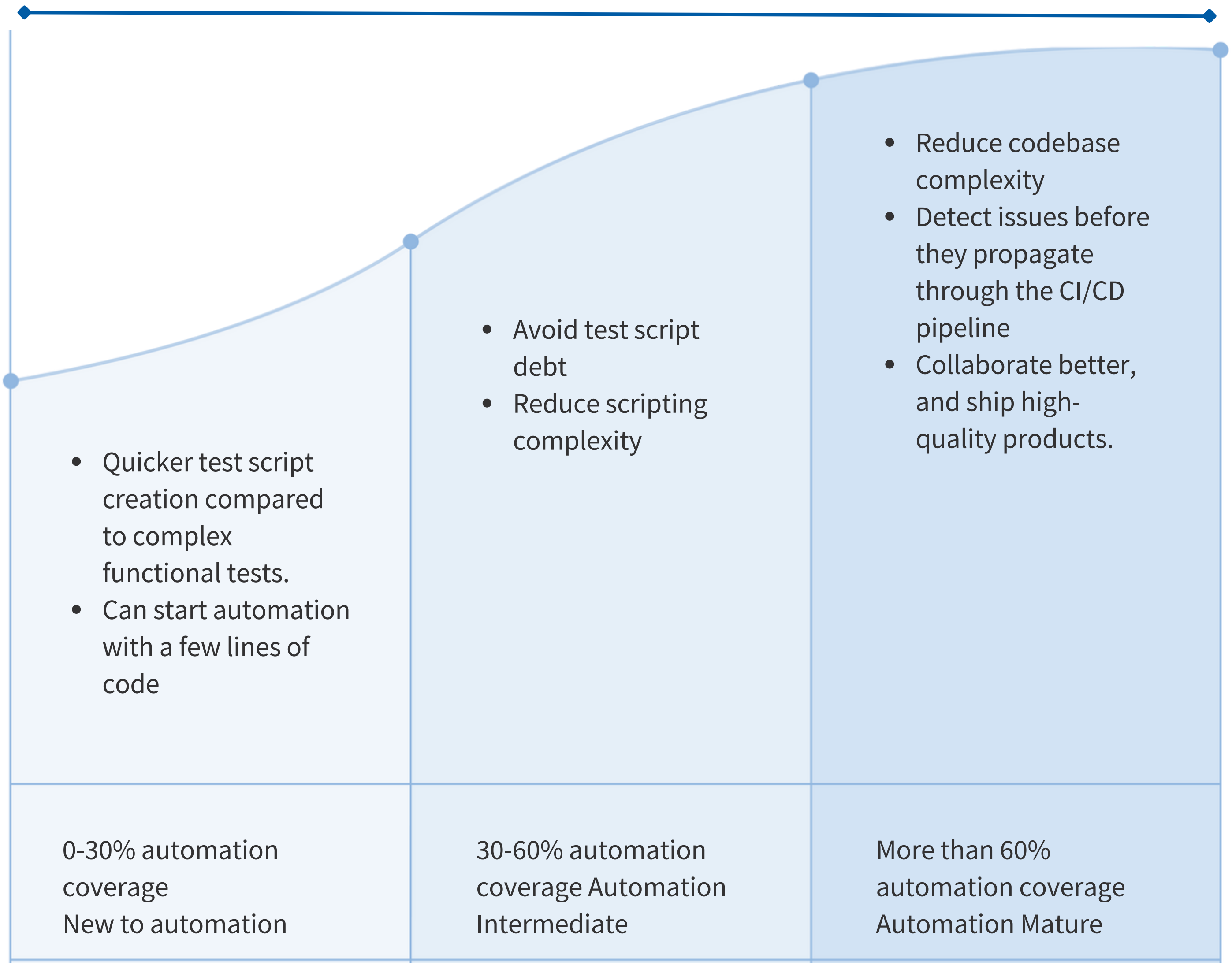
Aspect	Visual-First Automation	Functional-First Automation
Code Efficiency	Emphasizes writing less code for functional validation.	Requires eliminating all codes related to visual validation at the time of implementing automated visual testing.
Consistency	Guarantees cross-browser and cross-device consistency in the visual output.	Might succeed on one browser/device, but lacks assurance of consistent visual output across different platforms.
Collaboration	Enhances collaboration through user-friendly screenshots, easily understood by non-technical team members.	Being technical, it may lead to discrepancies in understanding among team members.
Test Coverage	Provides broad coverage of application appearance with relatively fewer test cases.	Demands more test cases for functional coverage, with requirements growing exponentially for a broader scope.
Bug Visibility	Visual bugs are visible to 100% of users, allowing for immediate feedback.	Functional bugs are only seen by users who encounter the specific functional issue.

Automated Visual Testing for Different Test Maturity Stages

Teams should prioritize and invest in maturing their testing operations if they want to generate the kinds of results that can make a real difference to their business. Generally, mature teams automate more tests, ensuring broader coverage and quicker releases, ultimately boosting customer satisfaction and revenue. Adopting a Visual-First approach, as opposed to a functional-first approach, offers advantages throughout the test maturity journey and helps you achieve test maturity faster.

In the early stages (teams with 10-20% automation), it simplifies automation as you have to write much less code. For intermediate teams (50-80% automation), it seamlessly integrates into existing tests, reducing script complexity. In mature teams (>80% automation), it's a game-changer, catching visual bugs early in CI/CD pipelines and enhancing collaboration for high-quality product delivery.

Why use Visual-First Automation



- Quicker test script creation compared to complex functional tests.
- Can start automation with a few lines of code

0-30% automation coverage
New to automation

- Avoid test script debt
- Reduce scripting complexity

30-60% automation coverage Automation Intermediate

- Reduce codebase complexity
- Detect issues before they propagate through the CI/CD pipeline
- Collaborate better, and ship high-quality products.

More than 60% automation coverage
Automation Mature

Introducing BrowserStack Percy and App Percy

BrowserStack's Automated Visual Testing platforms, Percy and App Percy, offer efficient and comprehensive solutions for handling the end-to-end Visual Testing process. With these platforms, developers gain full visual confidence, enabling them to build and maintain visually-perfect products more productively.

Here's how Percy and App Percy can transform your Visual Testing:

Zero Code Change Integration

Eliminating the need for manual code changes in your test scripts, you can easily incorporate the BrowserStack SDK into your existing test scripts, provide authentication details, and gain instant access to Percy and App Percy.

Rapid Build Execution

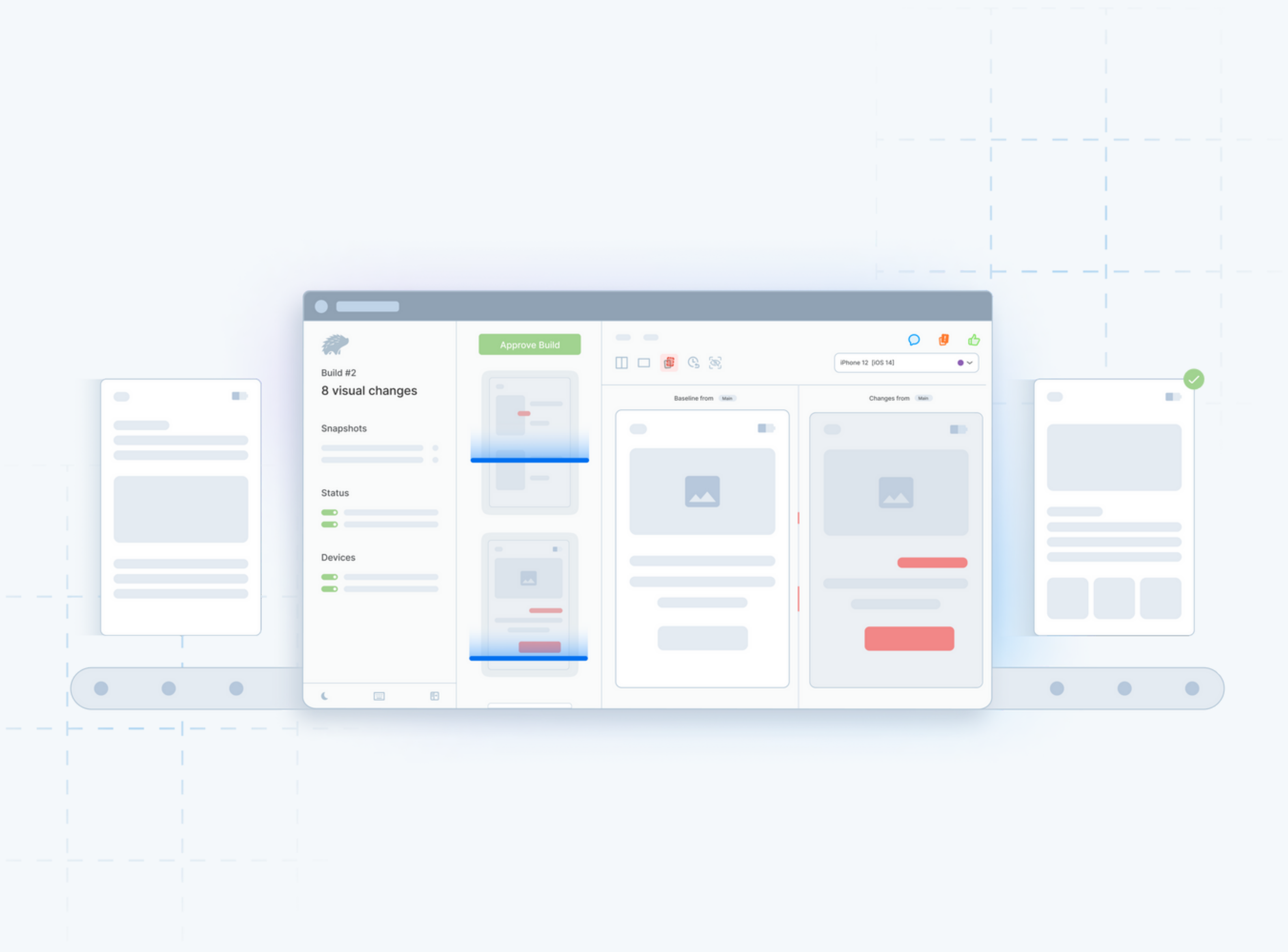
With Percy and App Percy, you can experience the power of running over 90% of builds in under 2 minutes. Leveraging DOM Snapshotting and advanced parallelization, these platforms ensure faster releases and pixel-perfect pages. The utilization of advanced computer vision algorithms facilitates the execution of builds on over 20,000 real devices across 19 data centers, boasting a 99.9% uptime guarantee.

Efficient Collaboration

Team collaboration becomes more accessible with Percy and App Percy. These platforms group visual changes and filter out noisy elements for faster and clearer reviews. Automatic status updates in pull requests keep the entire team informed about detected visual changes and updates throughout every visual review.

"App Percy has recently unlocked an entirely new avenue of testing for mobile native apps as well with visual comparisons. Functional testing is still needed of course, but it doesnt cover subtle UI differences that could be bugs."

James V. Source: G2



Try Percy and App Percy Today!

Get Started **Free Trial**