nagarro

Beyond green tests:

Extending quality engineering with Vision AI

2026

Whitepaper

Executive summary

Quality engineering has evolved significantly over the past decade. Modern automation frameworks now allow teams to validate workflows, business rules, integrations, and system behavior at unprecedented scale. Yet many of the issues that ultimately shape user experience still fall outside the reach of traditional functional testing.

An application can pass every automated test and still be released with broken layouts, inconsistent branding, localization errors, accessibility gaps, or visual deviations from the approved design. These are not failures of automation. They are signs of a broader quality gap: traditional automation is highly effective at verifying that software works, but less effective at evaluating whether the delivered experience looks, feels, and communicates as intended.

Recent advances in Vision AI and multimodal models create a practical opportunity to address this gap. By enabling automated systems to evaluate screenshots, interfaces, and visual experiences with more human-like reasoning, Vision AI introduces an additional layer of quality assessment that complements existing automation investments. Existing tools such as Selenium, Playwright, BrowserStack, and CI/CD pipelines can continue to handle navigation, interaction, and state validation, while Vision AI evaluates the rendered experience against approved baselines, design intent, brand standards, accessibility expectations, and market-specific requirements.

Drawing on real-world implementations across healthcare, manufacturing, aviation, and online gaming, this paper explores how organizations are using Vision AI to solve challenges that have traditionally required manual review. These implementations addressed diverse quality challenges including responsive validation across devices, localization consistency across global websites, Figma-to-production design fidelity, continuous compliance and brand monitoring, and automated testing of Canvas and WebGL-based applications.

Key areas covered include:

- Responsive experience validation across devices and screen orientations
- Global website and localization consistency assessment
- Figma-to-production design fidelity validation
- Continuous quality, compliance, and brand monitoring
- Automated testing of Canvas and WebGL-based applications
- Architectural patterns and implementation best practices
- Adoption considerations, challenges, and lessons learned

Who this paper is for

This paper is intended for quality engineering leaders, test automation teams, digital product owners, engineering leaders, and organizations responsible for delivering high-quality digital experiences across markets, devices, brands, and regulatory contexts.

It will be especially relevant for teams that already have mature automation in place but still rely on manual review to assess visual consistency, responsive behavior, localization, design fidelity, or interfaces that cannot be reliably tested through DOM-based automation.

© 2026 Nagarro

Table of Contents

© 2026 Nagarro

The gap between green tests and good products

There is a moment every quality engineer knows. The pipeline finishes, every test passes, the release goes out, and within hours, someone sends a screenshot. A layout is broken. A button is missing. An image hasn't loaded. Something that no test caught has reached a real user.

This isn't a story about bad automation. It's a story about the limits of what traditional automation was ever designed to evaluate. Our test suites have become extraordinarily capable at verifying that code behaves as specified. They are structurally less capable at evaluating the quality of the delivered experience, whether a page communicates clearly, whether a layout remains usable across devices, or whether the design intent survived the journey from Figma to production.

The gap between 'all tests passed' and 'the product is actually good' is where most of the interesting quality problems live. Closing it requires a different kind of capability, one that can look at a rendered page the way a user would, rather than reading the DOM the way a machine does.

Vision AI, in the form of large multimodal models capable of analyzing images and reasoning about what they contain, is emerging as the most promising way to bridge that gap. Not as a replacement for the automation practices that QE teams have built, but as an extension that brings a genuinely new kind of evaluative power to the testing process.

© 2026 Nagarro

What we ask of automation, and what we do not

Modern automation frameworks are remarkable. A well-built suite can execute thousands of assertions across dozens of browsers in the time it takes to make a cup of coffee. It can simulate user flows, validate API responses, confirm database state, and trigger alerts when anything deviates from expectation.

And yet the quality problems that reach users tend not to be the ones that automation is good at catching. They are subtler, more visual, more experiential. Understanding why requires being honest about what traditional automation was built to do, and where its architecture makes certain categories of evaluation structurally difficult.

This is where Vision AI becomes valuable – not as another assertion layer, but as an experience-evaluation layer.

Functional assertions verify existence, not experience

A test that confirms a button is present, has the correct label, and triggers the expected action when clicked is doing exactly what it was designed to do. It is not designed to evaluate whether that button is visually prominent enough to find, whether it is buried below the fold on a 375px screen, or whether a recent layout change has placed an overlapping element in front of it in one specific browser. The automation passes. The user experience fails.

Existence and experience are different properties. Automation has always been excellent at the former. The latter requires something closer to perception: the ability to look at a page as a whole and assess whether it works as intended.

For example, consider a checkout flow where the “Place Order” button renders correctly in the DOM — right label, right click handler, assertion green. A cookie-consent banner loads a fraction of a second later and overlays it on a 375px screen. Functionally, nothing is broken: the button exists and responds to a programmatic click, which is all the assertion checks. Experientially, the button is unreachable, the shopper sees a banner covering the only path to purchase. Revenue drops in one browser, on one breakpoint, and every test stays green, because no assertion was ever looking at whether the button was actually visible to a human.

Pixel comparison catches changes, not problems

Screenshot comparison tools were introduced to address this gap, and they are effective at detecting many types of visual regressions. Their fundamental limitation, however, is that they treat every visual difference as equally significant. A hero image that renders at a slightly different compression level, a font that anti-aliases differently on Windows versus macOS, a carousel mid-rotation, a live timestamp: all produce diffs that are technically correct and practically meaningless.

Teams respond by increasing thresholds, adding exclusion zones, and tuning sensitivity until the tool stops alerting on noise. In the process, they often also reduce sensitivity to real problems. The signal-to-noise problem in pixel comparison is not a failure of implementation. It is a consequence of treating visual evaluation as a mathematical operation rather than a qualitative judgment.

© 2026 Nagarro

What we ask of automation, and what we do not

Some interfaces have no DOM to query

A growing category of digital experiences, including games, rich data visualizations, 3D product configurators, interactive maps, and canvas-based applications, render their interfaces directly as pixels using Canvas or WebGL. There is no element to locate, no attribute to read, or no event to intercept. Traditional DOM-based automation simply cannot interact with these interfaces, which means they are either tested manually, tested inadequately, or not tested at all.

Design intent does not survive in assertions

When a designer specifies a particular shade of brand blue, a specific line height, a deliberate amount of whitespace around a call to action, none of that intent is represented in the test suite. The test checks that the element exists and is visible. It does not check that it looks right. Over many releases, this gap compounds quietly. Each small deviation from design intent accumulates until the product looks like a rough approximation of the brand it was supposed to represent, and no test ever flagged it.

Functional automation	Pixel comparison	DOM-based tools
Excellent at verifying behavior, state, and structure. Not designed to evaluate visual quality or experiential clarity.	Catches layout shifts and rendering changes. Struggles with dynamic content, cross-platform variation, and distinguishing signal from noise	Powerful for web interfaces. Completely blind to Canvas and WebGL rendered experiences where no queryable structure exists.

© 2026 Nagarro

What happens when your tests can see

Vision AI, specifically large multimodal models capable of analyzing images and reasoning about what they show, does not replace the automation infrastructure teams have built. It extends it at the evaluation step. Existing scripts continue doing what they do well: navigating, authenticating, reaching the correct application state, capturing screenshots. Vision AI takes over from there, treating those screenshots not as pixel arrays to compare, but as visual experiences to interpret.

The distinction matters. Comparison asks: is this different from the baseline? Evaluation asks: does this page communicate what it is supposed to communicate? Is the hierarchy preserved? Is the call to action findable? Does the layout hold up on this device? Has the design intent survived the implementation?

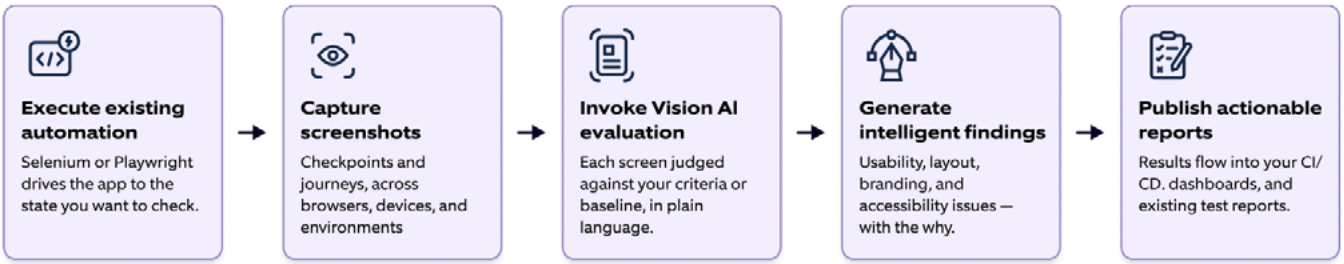
These are questions that can have nuanced answers, and that can be specified in natural language. Rather than defining expected pixel values, teams specify quality intent in plain language: confirm the footer includes the required legal notice, verify the product image matches the locale-specific variant, or ensure the visual hierarchy clearly distinguishes headings from body text. Vision AI applies those criteria consistently across hundreds of pages, in the time it would take a manual review team to check a handful.

Where it integrates.

The integration point is simpler than most teams expect. Automation captures the screenshot; Vision AI evaluates it. The two layers run in sequence, and the existing framework does not change. We have plugged this evaluation layer into Java Selenium suites, Playwright pipelines, and BrowserStack integrations without requiring any migration or infrastructure change. The vision capability is additive.

What changes is what the test run can tell you. Not just 'this element is present' but 'this element is visually accessible, contextually appropriate, and consistent with the approved design.' Not just 'these pages differ' but 'here is what differs, why it matters, and which component is responsible.' That shift in output quality is what changes how teams interact with test results.

Typical Vision AI workflow



© 2026 Nagarro

Five contexts, five different arguments for Vision AI

Each engagement offered valuable lessons about where Vision AI excels, where evaluation criteria require careful design, and which quality issues only become visible when viewed through a different lens. We have tried to share not just what we built, but what we actually learned.

1. Global website consistency at launch scale | Electronics manufacturing

The situation	What we built
A global brand was rolling out localized versions of its website across twelve regional markets. The UK site was approved for reference. The question was whether the French, German, Japanese, and other variants told the same brand story: the same visual hierarchy, the same product tile structure, the same hero section behavior. At 200+ pages per market, manual QA was not a strategy. It was a bottleneck that would delay the launch and introduce inconsistency through reviewer fatigue. And there was a specific technical complication: hero sections with autoplay video. Any comparison tool that compares two pages with autoplay heroes directly would generate continuous false positives, because the video frames are different by definition.	We built a CSV-driven comparison utility where each row contained a baseline URL and its regional counterpart. Playwright launched both URLs simultaneously across browsers and device sizes, capturing screenshots under identical conditions. Before comparison, the utility detected video elements on each page and paused them immediately after page load, capturing the first stable frame. Vision AI then analyzed each pair for semantic alignment: not asking whether pixels matched, but whether the regional variant told the same story as the baseline: same layout logic, same brand hierarchy, same product presentation, content translated correctly for the market. Heatmap overlays visualized structural drift, and the report highlighted specific tiles and banners requiring attention.

Outcomes

- **200+ pages across global markets** validated automatically per run, with manual review focused only on flagged differences, reducing QA effort by over 80% compared to a full manual review cycle.
- **Video false positives: eliminated.** The pause-and-capture approach converted the most unpredictable source of comparison noise into a non-issue. This felt like a minor engineering detail; it turned out to be foundational.
- **Genuine localization issues surfaced.** The tool flagged a product tile in the Japanese variant using an image approved for the UK market, a navigation element rendering with different font weight in German, and a banner text overflow issue in French that only appeared at specific viewport widths. None of these would have been caught by a functional test suite.

© 2026 Nagarro

Six dimensions vision AI evaluated per page pair

Content & messaging	Semantic equivalence of text: not translation checking, but meaning preservation
Layout & structure	Section ordering, element positioning, and responsive behavior consistency
Brand & visual identity	Logo placement, color usage, typography weight and hierarchy
Product assets	Correct locale-specific images, pricing format, currency symbols
Navigation & wayfinding	Menu structure, link availability, breadcrumb consistency
Dynamic content handling	Video, carousel, and animation behavior consistently captured

2. Figma to production design fidelity | Aviation

The situation	What we built
Design drift is insidious because no single deviation is large enough to trigger a conversation. A spacing value is rounded during implementation. A font weight that shifted in a CSS update. A button using the hover-state color as its default. Individually, each change is minor. Collectively, across a site with hundreds of pages and multiple releases per month, they compound into a product that looks like an approximation of the brand it was supposed to represent. The airline's QA team was stretched thin across functional testing. Synchronous design review sessions happened infrequently, caught only the most visible regressions, and created friction between design and engineering teams who were working from different mental models of what 'right' looked like.	We built a utility that compared Figma design exports directly against browser screenshots of the live application, evaluated across six specific dimensions. Rather than a binary pass/fail, Vision AI assessed each dimension on a gradient: how far had the implementation drifted from the design intent, and in which specific area. Block-level comparison broke pages into component-sized sections and scored each independently, so a page that was nearly perfect in the body but significantly off in the hero section would surface that specific deviation rather than averaging it away in a page-level score. Dynamic regions such as carousels, video, and animations were intelligently suppressed. The integration was built directly into the team's existing Java Selenium framework without any migration.

© 2026 Nagarro

Six dimensions vision AI evaluated per page pair

Layout & spacing	Section ordering, element alignment, whitespace: does it match the spatial logic of the design?
Typography	Font family, weight, size, line height, and visual hierarchy between heading levels
Color & brand compliance	Brand color adherence and contrast ratios for both aesthetics and accessibility
Component rendering	Buttons, forms, cards, icons: are reusable components rendering as specified?
Content accuracy	Missing content, placeholder text visible in production, truncation and overflow issues
UI drift score	SSIM and perceptual hash scoring at block level to quantify and localize deviations

What we did not expect

- **Block-level scoring was the most actionable output.** A page-level similarity score of 87% tells a developer nothing about where to look. Knowing that the hero section scores 71% while the footer scores 98% tells them exactly where to start. This specificity was the feature that got the most positive feedback from the engineering team.
- **Selenium integration was simpler than anticipated.** The vision evaluation layer sat cleanly alongside existing Selenium scripts. Teams kept their navigation logic entirely intact and simply added the evaluation call after each screenshot capture. No migration, no disruption to the existing test pipeline.

© 2026 Nagarro

3. Responsive validation for elderly users | Healthcare

The situation	What we built
The platform served a predominantly elderly user base completing high-stakes tasks: ordering medications, reviewing dosage information, scheduling care. Pages were breaking on certain device configurations: layouts collapsing, buttons disappearing below the fold, text truncating in ways that changed its meaning. The existing automation confirmed that every element was functionally present and clickable on the test device. It said nothing about how those elements appeared or whether they appeared at all, on the range of devices real users were actually bringing to the experience.	We built an evaluation model based on approved baselines rather than pixel matching. After navigating each critical flow across every target device configuration, screenshots were captured and reviewed by a QE who approved them as the semantic contract for that page and device. 'Semantic contract' is important: we were not locking in pixel values; we were locking in the intent, the hierarchy, the visual prominence of actions, the layout logic. Subsequent deployments were evaluated by Vision AI against that contract, assessing whether the intent was preserved rather than whether the pixels matched. Portrait and landscape orientations were tested as independent scenarios.

1. What this taught us

- **Portrait and landscape are not the same test on a rotated screen.** The layout adapts, content priority changes, and the user context shifts. Testing them independently caught an entire category of regressions that combined testing had been missing, including one case where a critical order confirmation step was pushed below the fold in portrait mode on a specific tablet size and had been invisible to testing for multiple releases.
- **Semantic baselines outlive design iterations.** Because the baseline captured intent rather than pixels, it remained valid across minor visual refreshes that changed colors or spacing without changing the layout logic. We updated baselines significantly less often than we expected to.
- **The report description matters as much as the detection.** When Vision AI identifies a regression, the quality of its explanation is just as important as the detection itself. The description of what changed, why it matters, and where the issue occurs determines whether a developer can take immediate action or must spend additional time investigating the problem. Investing in well-designed evaluation prompts consistently reduced triage effort by producing clearer, more actionable findings.

© 2026 Nagarro

4. Continuous quality monitoring across hundreds of pages | Healthcare Technology

The situation	What we built
A global medical technology company was supporting multiple franchises launching and maintaining their own websites across different regions and business units. While all websites needed to comply with corporate brand, legal, accessibility, SEO, and launch-readiness standards, each franchise site could differ significantly in structure, design, content, navigation, and product offerings. This created a challenge for quality assurance. Traditional automation frameworks rely on consistent page structures and reusable locators, but the variability across franchise websites made a single automation solution difficult to implement and maintain. As a result, launch validation was largely checklist-driven, requiring teams to manually verify dozens of quality, compliance, technical, and content-related requirements before a site could go live. The organization needed a scalable way to assess website readiness while accommodating the flexibility required by individual franchises.	Rather than attempting to automate individual page elements, we translated the organization's launch-readiness checklist into a set of independent validation modules powered by Vision AI and browser automation. Using Playwright and Stagehand as the navigation layer, the solution autonomously traversed each website, captured page states, and evaluated them against the agreed launch criteria. The system combined visual analysis, content verification, technical checks, accessibility validation, and compliance monitoring to assess whether a website met the organization's readiness standards. Each validation area was implemented as a modular check, allowing the framework to be applied consistently across websites regardless of differences in design, layout, navigation structure, or product catalog.

Reports were generated at two levels:

- **Executive summary reports** providing an overall launch-readiness status across validation categories.
- **Detailed implementation reports** highlighting specific findings, affected pages, and recommended actions for website owners.

© 2026 Nagarro

14 Validation modules: What each one checks

Page availability & load Pages load correctly, critical assets accessible, no 4xx/5xx errors	**Cookie consent** Banners present and functional where regulations require them
Accessibility Image ALT text, ARIA labels, and key accessibility requirements present	**Link validation** No broken, invalid, or unintended redirect chains on internal or external links
Footer compliance Required legal notices, compliance links, and regulatory content present	**Visual & layout integrity** Page structure, rendering consistency, and visual correctness
Brand compliance Naming conventions, trademark symbols, and brand language used correctly	**Product content** Product images, descriptions, and assets rendering correctly
Analytics verification GA4, GTM, and other tracking implementations firing as expected	**Robots.txt & sitemap** Crawlability correctly configured, sitemap present and valid
Domain & connectivity Domain accessibility, redirects behaving correctly, subdomain consistency	**SEO & metadata** Page titles, descriptions, schema markup, and JSON-LD structured data valid
Content consistency Vision AI check for placeholder text, missing content, and layout regressions	**Performance assessment** Core Web Vitals and page load metrics within acceptable thresholds

What this taught us

- **Checklist-driven validation scales better than page-specific automation** when websites vary significantly in design and content.
- **Modular validations are easier to maintain than large automation suites**, allowing new launch requirements to be added without reworking the entire framework.
- **Vision AI enables consistent evaluation across visually different websites**, reducing dependency on fixed page structures and locators.
- **Launch-readiness reporting creates a common quality standard** while still allowing franchises the flexibility to design and manage their websites independently.

© 2026 Nagarro

5. Autonomous validation of 1000+ slot games | Online gaming

The situation	What we built
The client managed a catalog of over 1,000 slot games from multiple providers. Each game required a tester to navigate the provider portal, launch the game, clear introductory screens, execute gameplay, and verify that spins, balance updates, and game states all functioned correctly, across browsers, continuously, as new titles were added and existing ones updated. The scale alone made this impractical for a human team. But the deeper problem was structural: slot games render on Canvas and WebGL. There is no DOM to query, no element to locate with a CSS selector, no attribute to read. The Spin button that a human tester can see and click instantly is invisible to a Playwright script attempting to interact through the DOM. This category of interface had historically been untestable by automation.	We built a distributed agent system. Games were loaded into a thread-safe execution queue, and multiple parallel agents pulled from the queue and ran validation workflows independently. Each agent used Vision AI to analyze game screenshots, identify the current state, whether loading screen, intro sequence, main gameplay, or bonus round, then determine the coordinates of interactive elements including the Spin button and balance display, and interact with them. The workflow: launch the game, navigate through intros, capture the pre-spin balance, execute a spin, capture the post-spin state, verify the balance change is mathematically correct. Cross-browser execution ran in parallel. Network throttling simulated degraded connectivity to profile load time behavior under realistic conditions.

Why this was only possible with vision AI

- **No DOM, no other option.** Canvas and WebGL render games as painted pixels. Traditional automation tools have no foothold. Vision AI analyzing the screenshot is not a workaround. It is the only technically viable approach for this class of interface.
- **Game interfaces have no shared conventions.** Each of 1,000+ games from dozens of providers has a different visual design. There is no consistent element ID to target, no common class name, no shared structure. Vision AI's ability to understand UI intent from visual appearance, recognizing that a circular button with a play icon is the spin control regardless of its specific visual design, is what makes this generalizable.
- **A new quality metric emerged that we did not plan for.** The system logged the number of attempts each AI agent required to locate the Spin button, originally as an operational metric to monitor agent performance. What emerged was a valuable indicator of interface quality. In well-designed games, the button was consistently identified on the first attempt. Games that required five or six attempts typically exhibited ambiguous visual hierarchies, poor affordance, or inconsistent layouts. This proved to be more than an AI performance metric; it became a proxy for usability. Unlike human testers, who intuitively adapt to interface inconsistencies and rarely document them explicitly, Vision AI exposed these patterns objectively and at scale. Notably, games with higher search attempts consistently aligned with player feedback describing the interfaces as confusing or difficult to navigate.

© 2026 Nagarro

Five architecture principles that held across all of it

These patterns emerged independently across engagements spanning different industries, clients, teams, and use cases. Their consistent recurrence suggests they are not isolated observations, but common characteristics of successful Vision AI implementations.

- **Approved baselines, not pixel thresholds** Every Vision AI workflow we built started with a human-reviewed and approved reference state. This shifts the quality model from 'does the code match the spec' to 'does the experience match what we have approved', a fundamentally more meaningful standard. Pixel thresholds are arbitrary; approved baselines are intentional.
- **Modular validation, not monolithic scripts**
  Separating concerns into discrete validation modules, covering visual, semantic, performance, and compliance checks, lets teams run targeted checks without the full suite every time, add new dimensions as requirements evolve, and diagnose failures with precision. Monolithic test scripts that check everything simultaneously are hard to extend and harder to debug.
- **Dynamic content suppression as a first-class concern**
  Autoplay video, carousels, animations, and live data widgets are the enemies of consistent visual testing. We learned to treat suppression strategies, including pausing video, freezing animations, and masking known-dynamic regions, as non-negotiable, not optional polish. Without them, false positive rates make results unactionable regardless of how good the evaluation logic is.
- **Parallelism at the agent level**
  Sequential testing at scale is a false economy. Thread-safe execution pools, as in the gaming validation, allow a single run to validate hundreds of items in the time a sequential framework processes a dozen. For large catalogs or global rollouts, parallelism is not an optimization. It is a requirement.
- **Reports designed for multiple audiences** The same test run produces different needs: executives want a category-level health summary; developers want specific failures with enough context to act; QA managers want trend data. Building multi-layer reports from the start, rather than retrofitting them later, determines whether findings get acted on or ignored.

© 2026 Nagarro

Key consideration for successful adoption

Vision AI has the potential to significantly expand the scope of quality engineering by enabling teams to validate visual, experiential, and contextual aspects of software that traditional automation cannot reliably assess. Realizing that potential, however, requires more than adding another tool to an existing test suite. It demands new ways of defining quality, designing evaluation criteria, and integrating AI into established testing workflows.

Across the diverse engagements presented in this paper, a consistent set of implementation patterns emerged, despite differences in industries, applications, and delivery teams. The following lessons are based on practical experience gained while applying Vision AI across diverse quality engineering scenarios and industries.

- **Baseline management is ongoing, not a one-time setup.** Baselines age. As products evolve, baselines must be updated deliberately. Otherwise, the gap between the baseline and the current intended state grows until the evaluation is measuring the wrong thing. Build a baseline review process into your release workflow from the
- **Evaluation criteria quality determines signal quality.** Vague criteria produce vague results. The specificity with which you describe what Vision AI should evaluate: what 'correct layout' means for this page, what 'brand compliance' means for this component, directly determines how actionable the output is. Investing in prompt design for the evaluation step is not overhead; it is the work.
- **Vision AI adds latency; design for it.** Evaluating screenshots through a multimodal model takes longer than a DOM assertion. For large test suites, this is meaningful. Parallel execution and selective application, running vision evaluation on high-value flows rather than every test, mitigate this without eliminating it.
- **The learning curve is real but bounded.** Teams need to develop intuition for what Vision AI catches well versus where DOM validation remains more reliable. The first few weeks of any deployment involved recalibrating expectations. The teams that pushed through that calibration period consistently found the system became more useful, not less, as they learned how to work with it.
- **Integration requires thoughtful architecture, not just an API call.** Calling a vision API and getting a response is straightforward. Building a system that integrates cleanly with existing automation, handles dynamic content reliably, generates actionable reports, and scales across a large test surface: that is an engineering project. Treat it accordingly.

© 2026 Nagarro

Where Vision AI fits – and where it does not

Vision AI is most valuable when quality depends on how an application is perceived rather than simply how it behaves. Existing automation frameworks continue to navigate the application, validate workflows, execute business logic, and establish the required application state. Vision AI complements this by evaluating visual, contextual, and experiential aspects of software that DOM-based assertions cannot reliably verify.

The two approaches solve different problems and work best together. Traditional automation verifies that the application behaves correctly, while Vision AI evaluates the quality of the rendered user experience.

Where Vision AI adds value:

Quality engineering challenge	How Vision AI helps
Responsive layouts	Validates layouts across devices, orientations, and screen sizes to identify broken or unusable interfaces.
Design fidelity	Compares the implemented UI against approved designs to detect unintended visual drift.
Localization	Verifies translated content, images, layouts, and regional branding remain consistent across markets.
Accessibility	Identifies visual accessibility issues such as missing labels, poor contrast, clipped content, or unreadable text.
Brand compliance	Ensures logos, colors, typography, and visual hierarchy remain aligned with brand guidelines.
Canvas and WebGL applications	Enables automated validation where traditional DOM-based automation has limited or no visibility.
Content-rich websites	Detects layout issues, missing assets, incorrect imagery, and visual inconsistencies that are difficult to validate using assertions alone.

© 2026 Nagarro

How to identify a good first pilot

Start with a small, high-value problem where manual visual review consumes significant effort or where traditional automation consistently leaves quality gaps.

Characteristic	Why it is a good candidate
Critical customer journeys	Small visual issues can directly impact user experience, conversions, or business outcomes.
Pages requiring frequent manual review	Reduces repetitive QA effort while improving consistency across releases.
Responsive or multi-device experiences	Captures layout issues that are difficult to validate through DOM-based automation.
Applications with frequent UI or branding changes	Detects unintended visual regressions before they reach production.
Multi-language or regional websites	Verifies localization, translated content, images, and regional branding at scale.
Canvas, WebGL, dashboards, or image-heavy interfaces	Provides automation where traditional locators are unavailable or unreliable.

Start small, then expand

Rather than evaluating every screen, begin with a single business journey or a limited set of pages. Define clear evaluation criteria, measure outcomes such as reduced manual review effort or additional defects detected, and use those learnings to expand incrementally.

© 2026 Nagarro

Closing thoughts

Sharing this feels important to us because the gap between ‘all tests passed’ and ‘the product is actually good’ is one that many teams have learnt to live with. It is the known unknown they compensate for through manual review, late-stage checks, and hope.

Our experience suggests that this gap does not have to be accepted as permanent.

Every engagement described here started with a team that already knew where traditional automation was falling short. The healthcare team knew that their elderly users were struggling on specific devices. The aviation team knew design intent was being lost between Figma and production. The gaming team knew validating a thousand Canvas and WebGL-based games manually was not sustainable.

Vision AI did not give these teams a new testing philosophy. It gave them a practical way to extend the quality engineering practices they had already built.

That is the practical lesson: if your testing strategy validates that your application works but not whether it looks, feels, and communicates as intended, there is a quality gap worth examining. The tools to start closing that gap exist today and can integrate with the infrastructure most quality engineering teams already have.

The best place to begin is usually not the entire test suite. It is one high-value journey, market rollout, page family, or interface type where visual quality still depends heavily on manual review.

© 2026 Nagarro

Authors

Deepshikha

Deepshikha leads Nagarro's AI in Testing practice and brings over 13 years of experience at the intersection of software quality and emerging technologies. She specializes in designing AI-driven testing strategies that enhance reliability, accelerate delivery, and help organizations build systems that are not only efficient but also trustworthy and resilient when it matters most.

Himanshu Kumar

Himanshu is an Automation AI Engineer in Nagarro's AI in Testing practice, where he drives key initiatives at the intersection of artificial intelligence and software quality engineering. Specializing in agentic frameworks, vision-based analytics, and self-healing automation, he focuses on architecting intelligent testing ecosystems that elevate software reliability and accelerate delivery.

© 2026 Nagarro