THE UNIFIED DIGITAL FRAMEWORK

The modern enterprise faces a critical paradox: while individual technologies evolve at breakneck speed, integrating these tools often lags, creating fragmented systems. Our approach dissolves these silos. By integrating Data Science, Full Stack Development, DevOps, and QA into a singular "Unified Digital Framework," we transform experimental technology into production-grade assets that drive measurable ROI for your organization. In the current digital economy, competitive advantage is no longer determined by access to technology alone, but by an organization's ability to scale its implementation. Businesses have been investing excessively in AI, cloud platforms, modern applications, and automation, yet many struggle to convert these investments into tangible business outcomes.

As technological ecosystems expand in digital areas, the organizations face growing complexities. The fragmented data sources, inconsistent data quality, and time-consuming manual analysis made it difficult for Data science teams to handle. Fullstack development teams struggled with applications tied to tightly coupled systems and complex integrations. The manual infrastructure management, environmental inconsistencies, and error-prone deployment process put pressure on the DevOps teams. Meanwhile, QA teams grappled with difficulty in validating expanding applications across various platforms and devices, often dependent on manual testing, leading to slower launches. Together these challenges create severe bottlenecks demanding solutions that not only solve individual technical problems but also connect workflows across disciplines to create a more unified, scalable, and resilient digital ecosystem.

Endure works towards building cutting-edge systems that are involved in providing end-to-end solutions to all the possible bottlenecks by helping to build and scale digital products across Data Science, Full-Stack Engineering, DevOps, and Quality Assurance. The Data Science service focuses on building intelligent AI systems, retrieval-augmented generation (RAG) workflows, evaluation frameworks, and autonomous monitoring solutions that improve the accuracy, scalability, and reliability of AI-driven experiences.Modern full-stack development involves the company building high-performance web applications, enterprise platforms, dashboards, and multi-brand ecosystems using modern technologies and scalable software architectures on both sides- frontend and backend. In addition, Endure’s DevOps services aim to establish reliable, resilient, and cloud-native infrastructure through automation, Kubernetes orchestration, observability, and disaster recovery strategies. Quality Assurance ensures product reliability through reliable automated testing that validates both expected user journeys and edge cases, ensuring faster releases, reduced risk, and consistent high-quality user experiences. By bridging the gap between innovation and execution, Endure aims to enable organizations to transform technology investments into expandable and measurable business outcomes.

AI-Driven Automation

Leveraging the Playwright Model Context Protocol (MCP) to enable AI agents to interact with the live application in a real browser session, navigating authenticated pages, inspecting the actual DOM and accessibility tree, and discovering accurate element selectors. This eliminates the traditional manual process of selector hunting through source code or browser DevTools. Instead, the AI agent explores each page programmatically—identifying buttons, headings, form fields, dialogs, and table structures—and maps them to resilient, accessibility-first locators. This MCP-powered discovery workflow was successfully applied to generate test suites for multiple frontend pages, with each generated spec using selectors verified against the actual production UI, resulting in tests that pass on first execution with minimal manual correction.

Adopted an AI-skill-based approach by creating a project-specific SKILL.md that encodes the testing team's house conventions—authentication patterns, fixture usage, cleanup strategies, selector priority hierarchy, and naming conventions. This skill file is complemented by a curated set of 14 reference documents sourced from the Currents.dev Playwright Best Practices Skill, covering locator strategies, assertion and wait patterns, fixture hooks, form validation testing, flaky test diagnosis, debugging workflows, and CI/CD integration patterns. This skill architecture enables any AI agent—whether Claude Code, Gemini, or VS Code Copilot—to generate tests that match the existing codebase style from the first draft, without requiring extensive manual rewriting. This approach lays the foundation for future self-healing capabilities, where the same skills will guide an AI agent to automatically diagnose and fix broken tests in CI/CD when frontend components are refactored.

Comprehensive Coverage

Building a robust Playwright-based testing framework that validates critical user journeys, including project creation and lifecycle management, app deployment, agent configuration, access token management, and package workflows through fully automated browser-driven tests. The framework integrates both frontend UI tests and API-level tests (authentication, session management, token refresh) within a unified Playwright configuration, using a structured project pipeline with cookie-based authentication, shared storage state, and environment-driven configuration. The API testing layer validates backend interactions, including login flows, token generation, session verification, and GraphQL query responses. By combining UI-driven validation with API-level assertions in the same framework, the focus is on maintaining a single regression suite that catches both frontend rendering issues and backend contract violations simultaneously.

Each test suite includes a deliberate balance of positive and negative scenarios. Positive tests validate the complete CRUD lifecycle, creating a resource with valid inputs, verifying its presence in the UI, interacting with it (editing, deploying, configuring), and deleting it with proper confirmation, ensuring the expected user flow functions correctly end-to-end. Negative tests systematically validate error handling by testing boundary conditions: submitting forms with empty required fields, attempting to create resources with duplicate names, uploading incorrect file types, and entering wrong confirmation text in destructive dialogs. This dual-validation approach ensures that both the happy path and edge-case failures are covered, strengthening overall test reliability and catching regressions early.

Reliability Benchmarking

Restructured the Playwright test architecture for long-term maintainability, scalability, and CI/CD readiness. This includes separating authentication fixtures from project provisioning fixtures, introducing a teardown-based cleanup pipeline that automatically removes test data after execution, implementing multiple reporting formats (HTML for visual inspection, JSON for machine-readable failure data, and PDF for stakeholder distribution), and organizing test files into logical domain-based directories covering authentication, navigation, project lifecycle, studio features, access management, and settings. The pipeline is structured with explicit project dependencies set up, project provisioning, test execution, and automated cleanup, ensuring each test run starts from a clean state and leaves no orphaned test data behind.

Created comprehensive documentation to support team-wide adoption and ongoing maintenance: a conventions guide defining the house style rules that both human-written and AI-generated tests must follow, a coverage gap analysis with a prioritized four-tier backlog for systematic expansion, an AI skill evaluation comparing three approaches (Currents.dev skill, Claude + MCP, and prompt-based generation) with a recommendation for a hybrid workflow, and a self-healing architecture proposal outlining how AI agents can automatically detect, diagnose, and fix broken tests in CI/CD pipelines. The architecture ensures the test suite is fully portable and executable across local development, staging, and CI/CD environments without requiring manual intervention or hardcoded credentials, enabling early detection of vulnerabilities and reducing long-term technical debt as the application scales.