Pendless

Pendless White Paper

Prepared by the Pendless Founding Team
November, 2025

Table of Contents:

Executive Summary

Pendless is a new kind of browser automation platform — one that turns plain English instructions into web actions in seconds. No infrastructure, no training, no maintenance headaches. Whether through a chat interface or an enqueueing API, Pendless lets users automate anything they can do in a browser — from filling out forms to collecting data — with the same precision as traditional code-based tools, but with none of their complexities.

At its core, Pendless introduces a queue-based automation model. Tasks can be enqueued, organized, and executed sequentially or programmatically through the Pendless API. This architecture makes automation predictable, traceable, and easy to integrate into existing systems. Each task runs directly within a browser context, ensuring compatibility with dynamic sites and real-world web behaviors.

Beyond automation, Pendless is designed as a conversation with the browser. Users can instruct it naturally and modify tasks through a chat-like interface that feels intuitive yet operates

with technical depth. Developers can extend the same capability through APIs that programmatically enqueue and manage tasks, allowing seamless integration with scripts, dashboards, and backend services.

Pendless was built to make web automation accessible and reliable. By unifying conversational control, task orchestration, and API-based extensibility, it lays the foundation for a new category of browser-native automation — one where the web becomes not just a place you visit, but a workspace you can program.

Problem Statement

In our decades of experience in IT development and delivery, we've seen a recurring, deeply human fear: being replaced by machines. As we enter this new AI spring driven by large language models, it's time to face a truth — yes, AI will remove jobs. But we must ask: *which* jobs? Ideally, the boring ones nobody wants to do. Ask any teenager what they aspire to be — none will say "data entry specialist."

In our view, much of the current discourse around AI is misdirected. While many efforts focus on replacing professionals like doctors or lawyers, or creatives like designers and musicians, the real opportunity lies in freeing humans from mechanical, repetitive work. Human activity should remain fundamentally human — enhanced by technology, not displaced by it. Since the industrial revolution, progress has always meant building tools that extend our abilities, not ones that replace our purpose.

That's the principle at the core of Pendless: AI should improve our relationship with machines, not distort relationships between humans.

Across our careers, we've watched organizations celebrate the latest AI demos while teams next door spent countless hours performing manual, repetitive browser tasks — filling forms, copying data, re-uploading files, reporting to agencies. Large enterprises have adopted Robotic Process Automation (RPA) tools to address this, but those systems are expensive, rigid, and fragile. A single change in a target website can break the entire automation pipeline, forcing costly rebuilds and delays — all under hefty license fees.

But what about small and medium businesses? These highly competitive, operations-driven teams still rely on human labor for browser-based workflows — tasks that are time-consuming, error-prone, and unfulfilling. Studies show that workers spend an average of **581 hours per year** on repetitive digital activities, and **60%** believe they could save at least six hours a week with proper automation. Yet only **20%** of SMBs have any RPA or automation solution in place — mainly due to cost, complexity, and poor ROI (Sources: McKinsey, Smartsheet).

Pendless

Pendless Vision

We built Pendless for the rest of us - an AI powered robot that lives in the browser and is ready to automate any web task expressed in plain English. It's fast, reliable, and ready to help businesses of any size automate everyday workflows without code, without setup, and without depending on external infrastructure.

Pendless embodies our mission: to empower companies to stay competitive with practical, human-centered AI that works. We believe automation should amplify people, not replace them— freeing teams from mindless tasks so they can focus on creative, strategic, and meaningful work.

Technical Architecture

At the beginning of our development journey, we faced several strategic decisions. Should we build a desktop robot? Fork our own version of Chromium? Or use a browser emulator to interact directly with web services? All these options contradicted our principle of keeping things simple. Instead, we chose to build a **Google Chrome Extension**, leveraging its code injection capabilities to interact seamlessly with web pages.
Pendless follows a four-layer architecture:

1. 1. Chrome Extension Side Panel. This serves as the user interface, organized into three tabs:
    - Chat – a conversation with the AI where users can ask questions, test, and organize automations.
    - Queue – a list of tasks ready to be executed, with options for automatic execution.
    - Gallery – a collection of saved or pre-tested prompts for future use.

2. Automation Engine. The engine injects automation code directly into the target page, allowing direct access to the DOM. Its dual role is to:
    - Understand the current page elements.
    - Execute automations reliably.

    This layer is critical: it bridges the gap between AI instructions and real-world browser actions, overcoming limitations inherent in dynamic web pages.

3. Server-Side Processing. Beyond standard tasks like user management and usage tracking, the server processes requests from the Automation Engine. It curates the DOM, prepares instructions for the LLM, and handles guardrails that stabilize the AI output. We benchmarked multiple approaches and decided to use OpenAI's completion API. Despite careful input stabilization, LLM outputs still require significant server-side processing to produce reliable automation instructions.

4. Enqueueing API. A simple yet robust API allows tasks to be enqueued on Pendless. Developers can submit a URL and desired action in natural language, then monitor status and feedback. Combined with an auto-start toggle in the Queue, this enables fully unsupervised, no-code automations directly in the browser.

   We've also added middleware origination points, like automated email readers or task launchers from folders, and we plan to expand this stack further.

Performance, Security and Reliability

Our measurements show that 90% of automation time is due to the LLM API latency, which we expect to decrease as model infrastructure and bandwidth continue to improve. A fully autonomous Pendless instance can currently execute complex tasks in roughly 30 seconds. This performance has important implications. By eliminating human reaction time, navigation delays, and the risk of manual errors, Pendless delivers not only faster results but also consistency at scale — every execution takes the same predictable path, with identical timing and precision. Tasks that might take a human several minutes can be completed in seconds, and repeated hundreds or thousands of times without degradation in accuracy or attention. In operational environments, this consistency translates to measurable gains in throughput, reliability, and overall process stability.

Security and reliability are central to our design. Communications and storage are encrypted, and by default, conversations are not stored due to potential sensitive content. We implemented multiple guardrails against hallucinations from the LLM. Our experience shows that the OpenAI API's temperature parameter alone is insufficient: too low, and the system cannot define automation steps; too high, and it may ignore our guardrails. Pendless is designed to execute exactly what it is told, relying on well-defined prompts while minimizing unintended behaviors.

Limitations

While Pendless is designed for reliability and versatility, certain constraints remain:

- Synthetic interactions: Some web pages implement protections that prevent simulated clicks or inputs, which may limit automation on highly protected sites.
- Iframes and embedded content: Automations cannot always access content inside cross-origin iframes due to browser security policies.
- Non-HTML content: Pendless currently cannot interpret or extract information from images or PDF documents.
- Dynamic rendering edge cases: Very complex single-page applications or heavily obfuscated DOM structures may require carefully crafted prompts or additional adjustments.

These limitations are inherent to the browser context and web security model. Understanding them allows users to design tasks within the scope where Pendless excels — interactive HTML-based web workflows.

Agentic vs. Robotic Behavior

A key design decision was defining Pendless's orientation. With clear prompts, Pendless behaves robotically, executing tasks predictably. When instructions are less defined, the AI exhibits full agentic behavior: it performs intermediate actions, monitors the results, and waits for feedback before completing the task. This allows the system to adapt to uncertain scenarios while still ultimately following the user's intent.

Development Insights

Learning from using Pendless ourselves

Throughout development, the team became natural users of Pendless. This experience reinforced a key insight: well-crafted prompts directly affect automation reliability. Saving tested prompts to the Gallery not only preserves successful automations but also improves efficiency over time. Iterating on prompts during early tests helped shape a user-centric interface and practical workflow patterns.

Prompt engineering and LLM behavior

Even after stabilizing inputs, the AI output required extensive processing to become a dependable automation instruction set. We learned that:

- Temperature settings alone are insufficient for reliability. Low temperatures limit the system's ability to define steps, while high temperatures break guardrails, making the system too imaginative.
- The AI exhibits random variations in output, which can impact stability. Addressing this became a full subproject, critical to ensuring consistent operations for users.
- Iterative conversations with Pendless helped us refine prompts, teaching us how to achieve precise, predictable automations. This hands-on learning remains valuable

advice for users seeking the most reliable results.

Stability and feedback loops
When instructions are incomplete or ambiguous, the system engages in full agentic behavior: performing intermediate actions, monitoring results, and waiting for feedback before completing a task. Understanding this behavior guided our approach to prompt design and system guardrails, balancing flexibility with control.

Security and reliability practices
Sensitive data handling shaped several design choices:

- Communications and storage are fully encrypted.
- Conversations are not stored by default to protect user data.
- Multiple filters and safeguards minimize hallucination risks, ensuring the system executes instructions as intended while adhering to predictable behavior patterns.

Middleware and automation triggers
Experiments with origination points, such as automated email readers or task launchers from folders, highlighted opportunities to expand Pendless beyond the core UI. These middlewares enhance usability, allowing tasks to start from diverse sources and improving workflow efficiency.

Key takeaway
The overarching lesson from our development journey is that stability, predictability, and reliability emerge from a combination of solid architecture, careful prompt design, and thoughtful feedback management. Pendless is designed to follow instructions precisely, while gracefully handling ambiguity when necessary — a balance that distinguishes it from traditional RPA or naïve AI automation solutions.

A curiosity

Early in development, we discovered that the most effective Pendless prompt builder was Pendless itself. We repeatedly asked the system how to craft the best prompts to generate reliable automation sequences, and in doing so, we learned as much from its suggestions as from our own experimentation.

Core Features

Pendless was designed around a single idea: browser automation should feel as natural as telling someone what to do. Every capability in the platform exists to make that possible —

combining the intuitiveness of conversation with the reliability of engineered automation. The result is a system that adapts to users rather than forcing users to adapt to it.

Conversational Automation Interface

At the heart of Pendless lies its chat-based interface. Users can simply describe what they want done in natural language — “fill this form,” “extract these records,” “summarize this page” — and Pendless translates those instructions into structured automation sequences. Each conversation becomes an evolving workspace where users can refine prompts, inspect results, and save improvements. This approach bridges human intent and executable action, eliminating the technical barrier that traditionally separates users from automation.

Queue-Based Orchestration

Pendless introduces a queue model for managing automations. Tasks can be enqueued, reordered, and executed manually or automatically, making processes predictable and traceable. This architecture provides a transparent workflow — users can always see what’s running, what’s pending, and what’s completed. It also allows Pendless to be integrated into more complex automation pipelines, enabling businesses to coordinate multiple browser tasks with precision and accountability.

Prompt Gallery and Knowledge

Experience showed us that better prompts lead to better automations. The Prompt Gallery was designed to preserve that knowledge. Users can save their best-performing prompts, reuse them across projects, or share them with teams. Over time, organizations can build internal libraries of reliable automations, transforming prompt engineering into an asset rather than a one-off experiment.

Enqueueing API and System Integration

Beyond the interface, Pendless exposes its capabilities through a lightweight API (see our GitHub page). Developers can enqueue tasks programmatically by sending a URL and an instruction in natural language, and then monitor execution and feedback. This API design allows Pendless to be embedded into scripts, dashboards, and backend systems — effectively turning it into a no-code, browser-native automation layer for existing workflows.

Agentic Responsiveness

When prompts are incomplete or ambiguous, Pendless doesn’t fail silently — it acts agentically. The AI performs intermediate steps, evaluates their outcome, and waits for feedback before completing the task. This balance between autonomy and control lets Pendless adapt to uncertain or dynamic web environments while preserving reliability.

Pendless

Use Cases

Pendless can be applied wherever repetitive browser work occurs, but its value becomes clearer when we categorize potential uses along four conceptual axes: Read-oriented, Write-oriented, Combination of Read and Write, and Web Application Interaction.

Read-Oriented Tasks

These automations focus on **extracting information from web sources**. Pendless can read content from websites, scrape structured data, monitor changes, or summarize text.

- Example: Collecting product prices across competitor sites, tracking stock updates, or aggregating news for market research.
- Value: Saves human attention for analysis rather than collection.

Write-Oriented Tasks

These involve inputting, submitting, or posting information into web interfaces. Pendless can fill forms, upload data, or post content across multiple platforms automatically.

- Example: Uploading inventory information to an e-commerce platform or submitting bulk reports to a government portal.
- Value: Reduces repetitive manual input, preventing errors and accelerating throughput.

Combined Read-and-Write Tasks

Many workflows require both reading and writing — for example, collecting data and using it to update another system. Pendless handles these multi-step automations seamlessly.

- Example: Monitoring customer feedback on multiple channels, analyzing sentiment, and updating CRM entries automatically.
- Value: Streamlines complex processes that are tedious and error-prone when done manually.

Web Application Interaction

Beyond standard forms and pages, Pendless can automate dynamic web applications, such as dashboards, SaaS tools, or internal web platforms. These tasks often involve conditional logic, multiple navigation steps, or interacting with single-page applications.

- Example: Performing automated QA in a web app, filling workflow templates, or interacting with multi-tab dashboards.

- Value: Extends automation to environments traditionally difficult to manage, with AI adapting to interface changes and user-defined goals.

By categorizing use cases this way, we illustrate Pendless's versatility while highlighting the core focus on browser-native tasks. Whether reading, writing, combining both, or interacting with complex web apps, the system transforms repetitive web workflows into reliable, predictable automations.

Comparison to Alternatives

Pendless exists in the broad landscape of automation tools, ranging from traditional Robotic Process Automation (RPA) platforms to modern no-code workflow builders. Each approach serves a different purpose — and understanding these boundaries highlights what makes Pendless unique.

Category	Pendless	Traditional RPA Systems	No-Code Automation Tools
Scope	Focused on browser-based automation — runs where humans already work: inside Chrome.	Broad system-level automation across desktop apps, legacy systems, APIs, and databases.	App integrations through prebuilt connectors (e.g., Google Sheets → Slack).
Setup	Lightweight, no install beyond a Chrome extension.	Heavy infrastructure, requires setup and orchestration.	Cloud-based, simple setup but limited to supported apps.
Interface	Conversational — automation is created through natural language prompts.	Technical, requires detailed scripting and mapping.	Visual block or form-based builder.
Execution	Runs in the browser, leveraging Chrome's DOM and user context.	Executes on virtual machines or desktop agents.	Executes via cloud services, often detached from user context.

Flexibility	Adapts dynamically to web changes, guided by AI interpretation.	Deterministic, needs explicit mapping for every step.	Limited to predefined app actions.
Integration	Through Pendless Queue and API.	Via enterprise connectors and orchestrators.	Through third-party integration libraries.
Target User	Knowledge workers, product teams, startups — anyone who automates browser work.	IT departments and large enterprises with cross-system needs.	SMBs or individuals automating simple app workflows.

Accuracy and Speed

Pendless conducted a head-to-head benchmark against Perplexity's Comet Browser Agent using an identical set of 34 automation prompts. The benchmark evaluated four core dimensions critical to browser-based automation performance: execution speed, precision, recall, and general error rate.

Across the 34 test prompts, Pendless achieved an average execution time of **23.4 seconds per task**, compared with **2 minutes and 13 seconds** for Comet. Precision — the percentage of steps correctly performed out of all attempted steps — reached **97.2 percent**, compared to Comet's 93.3 percent. Recall, measuring how many of the required workflow steps were actually completed, also favored Pendless by a wide margin. Pendless achieved **97.9 percent recall** versus Comet's 93.6 percent.

Future Roadmap

Pendless is designed as a living platform, capable of evolving with user needs, AI advancements, and the changing web. Our roadmap reflects both short-term enhancements and long-term strategic directions:

Expanded Middleware and Origination Points

We plan to grow the library of middleware connectors, enabling automations to be triggered from diverse sources — email, folders, webhooks, or other events. These origination points will allow businesses to integrate Pendless seamlessly into their existing workflows.

Third-Party Tool Integrations

To broaden utility, Pendless will seek deeper integration with popular SaaS platforms, productivity tools, and enterprise dashboards. By connecting directly to external systems, users will be able to orchestrate cross-platform workflows with the same natural-language simplicity.

Desktop Expansion

Now that we have stabilized LLM input and refined feedback mechanisms, the next step is moving beyond the browser. A desktop version of Pendless will extend automation to applications outside Chrome, bringing its conversational, queue-based orchestration to a wider range of workflows while preserving reliability and security.

Team Collaboration and Knowledge Sharing

We envision shared prompt libraries, team-level automation galleries, and analytics on task usage. These features will turn individual automations into collective organizational knowledge, fostering efficiency and consistency across teams.

Images and PDFs

We plan to extend Pendless's capabilities to support image and PDF content extraction. This will enable automations that currently require manual interpretation, further broadening the system's applicability while maintaining the platform's principles of reliability and natural-language-driven task execution.

Optional Local Relay for OS-Level Interactions that bypass certain restrictions

Some websites intentionally block synthetic browser interactions as a security measure, which limits what in-browser automation can do. One pragmatic path forward is an optional local relay (agent): a small, signed OS-level helper that users install and explicitly authorize. The Chrome extension would send authenticated, minimal click/input orders to the local relay over a secure channel; the relay would then synthesize native input events within the operating system (mouse/keyboard events, window focus), allowing interactions that are indistinguishable from real user input at the OS level.

Conclusion

Pendless represents a new approach to browser automation, combining natural-language instruction, queue-based orchestration, and AI-guided task execution. By operating directly within the browser, it enables reliable, repeatable automation of web workflows without the complexity or overhead of traditional RPA systems.

Pendless

The platform's design — including its Chrome Extension interface, Automation Engine, server-side processing, and enqueueing API — ensures that tasks are executed precisely, feedback is incorporated, and intermediate states are monitored for consistency. Lessons learned during development, particularly around prompt design, LLM output stabilization, and agentic behavior, have informed a system that balances flexibility with predictability.

Looking forward, Pendless is positioned to expand its capabilities: more middleware, deeper third-party integrations, autonomous multi-step execution, and eventual desktop support. Each development path is guided by the same principles of reliability, transparency, and efficiency, ensuring that automation enhances existing workflows rather than complicates them.

In sum, Pendless demonstrates that browser-native automation can be both technically robust and accessible, providing a platform where complex web tasks can be executed with minimal human intervention while retaining full oversight and control.