

# An Open-Source Browser-Based Master–Slave AI Agent Framework for Workspace Orchestration



Paper by  
**optimando.ai**



**Abstract:** We introduce a novel open-source framework, developed and published by **Oscar Schreyer and the AI consultation and automation integration company optimando.ai**, that transforms individual browser tabs into collaborative AI agents, enabling real-time orchestration and optimization of complex digital AI workspaces. The open-source system from **optimando.ai** helps users improve their digital work in real time by using a main browser tab (the master) for user activity and connecting it with AI-powered helper tabs (slaves) that analyze screenshots and inputs to give smart suggestions based on the user's goals; for example, if someone is building an IT job board and watches a video about n8n, the system can automatically suggest better GDPR-compliant workflows by using the business context and showing the results as text, visuals, or audio; the master tab can also be replaced with a camera input in future setups. In this system, a lightweight browser extension registers each tab with a unique ID and links it to a local orchestrator app. Users designate any tab as a master (primary interface) or slave (auxiliary agent) via the extension. Master tabs handle user interactions (e.g. with chatbots like ChatGPT, Claude, or Google Gemini, or browser apps and websites such as Google Sheets, YouTube, Amazon, email, news portals, or any other web-based service), while helper tabs

each run a distinct LLM chatbot (local or cloud) to perform supporting tasks. The extension captures only the browser’s visible content (screenshots and text inputs) from the master tab and routes it through the orchestrator to selected slave agent tabs. Slave agents can then do prompt optimization, tree-of-thought expansion, data retrieval, context augmentation, and more. All agents operate under “man-in-the-loop” control: the user supervises and approves outputs before use. Crucially, this system is open-source and browser-only, ensuring transparency, user control, and data sovereignty. We distinguish our approach from single-agent AI browsers (e.g. Opera Neon or Google’s Project Mariner) by enabling true multi-agent coordination across tabs, extensible to multi-monitor and team scenarios. We detail the system architecture, use cases, and ethical considerations, and outline the framework’s unique contributions and open-source vision.

Modern knowledge work often involves frequent switching between browser tabs and applications, resulting in cognitive overhead and productivity loss. Studies suggest users switch between digital interfaces over 1,000 times per day, often losing several hours weekly to simple reorientation.

Agentic AI systems seek to reduce this friction by acting as intelligent intermediaries across applications. Users can instruct such systems to retrieve data, automate steps, or manage multistep workflows across interfaces. Recent efforts such as OpenAI’s *Operator*, DeepMind’s *Project Mariner*, and Opera’s *Neon* browser illustrate growing capabilities in web-based agentic interaction using large language models (LLMs).

Architectures across these projects vary—ranging from browser-integrated assistants to cloud-hosted control environments. Common capabilities include form handling, navigation, summarization, and task execution

using multimodal input. While promising, deployment remains subject to broader industry challenges such as session continuity, transparency, and user-aligned control structures.

**The Optimando.ai framework introduces a modular, open-source orchestration concept** designed to operate within standard browser environments, optionally backed by locally hosted LLMs. It enables **real-time, context-driven optimization** using multiple **independent AI helper agents**, each operating in its own browser tab. These helper tabs may host **distinct LLMs such as the web-based versions of ChatGPT, Gemini, Project Mariner, Claude, Mistral, Grok, Llama and many more** selected by the user based on the task's privacy, cost, or reasoning complexity.

For example, lightweight or low-cost LLMs can be used in helper tabs handling repetitive or less critical tasks; advanced reasoning models can be reserved for complex, high-value workflows; and locally hosted LLMs may process sensitive or private information in fully self-contained tabs.

Input data to this system can be **multimodal**, including text, audio, screenshots, UI events, or camera streams. These triggers can originate from within the desktop browser or be activated remotely via smartphones or AR devices acting as **remote master agents**. Once toggled active, such devices send live input to a workstation.

Users may operate directly at the orchestration workstation or remotely initiate tasks from other locations. The system supports both **autonomous execution** and **human-in-the-loop workflows**, enabling oversight, corrections, or adaptive feedback. This flexibility allows real-time augmentation, background execution, and hybrid workflows tailored to user preferences.

**Input Data /  
Screenshots**



**Helper Tabs for  
Contextual Realtime  
Optimization  
with AI**



**AR Glasses**

automates web tasks (forms, shopping, etc.). Opera also demonstrates a more ambitious “AI engine” that, in the cloud, can work on user-specified projects offline and do multitasking in parallel. However, Opera’s agents are proprietary, deeply integrated into one browser, and not open for user modification. Opera One (a related product) has introduced AI Tab Commands: a feature where a built-in assistant can group or close tabs on command (e.g. “group all tabs about ancient Rome”). This helps manage tab clutter, but it still uses a single AI interface per browser to organize tabs, without supporting multiple cooperating agents.

Outside the browser domain, research on LLM-based Multi-Agent Systems (MAS) is rapidly growing. Tran et al. (2025) survey LLM-MAS and note that groups of LLM-based agents can “coordinate and solve complex tasks collectively at scale”. Emerging orchestration frameworks (e.g. AWS Bedrock’s multi-agent service or Microsoft’s AI Foundry) allow specialized agents to collaborate under a supervisor, and enterprises are experimenting with central “Agent OS” platforms that integrate many agents. But these systems operate at the level of backend services or applications, not at the level of coordinating a user’s browser environment. Crucially, we found no published work on orchestrating multiple AI agents distributed across browser tabs as a unified workspace.

**Privacy and Control.** As AI agents increasingly interact with web interfaces and user data, privacy and control have become central design concerns. Existing projects such as Opera's *Neon* emphasize that automation runs locally to preserve users' privacy and security. Similarly, OpenAI's *Operator* allows users to manually intervene in sensitive interactions via a "takeover mode" and is designed to avoid capturing private content without user intent.

The framework developed by Optimando.ai adopts a similar privacy-first philosophy. All orchestration components are **self-hosted** and run within the user-controlled environment. **No data is transmitted to any Optimando.ai server.** The coordination logic and agent communication infrastructure are open-source and designed to operate under user ownership and configuration. Users may choose to deploy the framework on local machines, private servers, or trusted edge devices depending on their needs.

While remote input streams (e.g., from smartphones or AR devices) may transmit data over the internet to the orchestrator, all transmission paths and endpoints are defined and controlled by the user. Optimando.ai **does not operate or provide any backend services** that receive, log, or process this data. The system's observation is also strictly limited to in-browser content in explicitly configured tabs. There is **no tracking of desktop activity or full-device behavior.**

Each helper tab in the browser may be assigned a task-specific AI agent, using either a locally hosted LLM interface or a third-party web-based LLM (e.g., ChatGPT, Claude, Gemini). The **choice of LLM, its operational mode (cloud or local), and any associated data sharing are entirely the responsibility of the user.**

Optimando.ai has no control over the behavior, data retention, or processing methods of any third-party services the user may connect.

The orchestration system itself is published as an **open-source project** under a permissive license (e.g., Apache 2.0 or MIT), and the conceptual framework described in this paper is released under the **Creative Commons Attribution 4.0 International License (CC BY 4.0)**. This dual licensing approach ensures both academic openness and practical reusability.

In summary, the architecture emphasizes **transparency, modularity, and user-level sovereignty**. It supports a wide range of deployment models while providing strong safeguards for those who require private, auditable AI automation without reliance on external orchestration infrastructure.

**Master/Slave Roles and Feedback Loop** Users can tag tabs as master (leading tasks, direct user focus) or slave (dedicated to background sub-tasks). These background tasks run automatically, triggered by activity in the master tab. Masters are often user-facing tasks (e.g. a chatbot or any browser-based application). Slaves provide assistance: for example, while a user debates a business strategy in a chat (master), one slave agent might track the user's goal progression, another might fetch relevant statistics, and a third might generate a relevant infographic.

For example, imagine a user completing a complex online tax declaration form in the browser (master tab). Prior to starting, the user can preload relevant information—such as financial records, identifiers, or past filings—into the system's global context to enable accurate real-time assistance. As the user progresses through the form, helper agents in background tabs analyze the visible content in real time. One agent might flag commonly misinterpreted sections, another could cross-check figures against preloaded values, and a third might suggest missing documentation typically required for the current section. The AI system understands the user's intent—to complete the form accurately and efficiently—and delivers targeted, context-aware suggestions along the way.

To ensure data privacy, users are advised to **anonymize personal information** before inclusion in the global context or to **assign sensitive fields to locally hosted language models** within secure browser tabs. The framework is designed to give users full control over how and where data is processed. This transforms the



browser into an **automatic real-time support**—a private, intelligent layer of guidance for high-stakes digital workflows.

Even more broadly, if a user watches a YouTube video (master tab)—for example, a product walkthrough of a new AI tool—slave agents can assist by cross-referencing this information with the user's current goals or projects. One agent might highlight how the showcased tool could be integrated into the user's existing tech stack. Another might suggest more efficient or better-suited alternatives based on predefined system constraints or preferences. A third agent could retrieve relevant use cases or success stories, helping the user assess practical value. Together, they act as a live research and recommendation engine—turning passive content consumption into actionable insights aligned with the user's broader objectives.

The user sees slave outputs as suggestions or context inserts in real time. This creates an optimization loop: the user steers the main task, slaves continuously augment it, and the user approves or refines the results. The human stays “in the loop” at every step, aligning with best practices in trustworthy AI.

**Technical and Ethical Considerations** The extension only captures the browser's visible content (screen images and user-entered text), not the full desktop. This guarantees all agent inputs are limited to in-browser context, protecting unrelated private data. Captured content is sent through the orchestrator to slave agents. No data is ever sent to outbound servers or external services unless explicitly configured by the user. Being fully open source, the software's code is transparent and available for public review, ensuring accountability and trust. The connected LLMs may communicate externally (e.g., web-based LLM chatbot), but this behavior is entirely under user control. Alternatively, local LLMs can be used based on the user's preferences or requirements.

# Feature Comparison

Feature	Proposed Framework	Opera Neon	Google Project Mariner
Realtime backend optimization suggester	Yes (context-aware AI suggestions tied to global user goals)	No	No
Multi-agent coordination	Yes (multiple slaves per tab)	Partial	Partial
Browser tab as agent	Yes (users tag tabs)	No	No
Open-source	Yes (user-run)	No	No
User-controlled LLM selection	Yes (any open or cloud LLM)	No	No
Data sovereignty	High (all local, opt-in cloud)	Medium	Low
Man-in-loop by design	Yes	Yes	Yes
Unique agent IDs (multi-user)	Yes	No	No
Multitasking / parallel tasks	Yes (multiple slaves)	Yes	Yes

# Evaluation and Distinction:

While some systems like Google's Project Mariner offer autonomous browser-based AI actions, they often lack persistent human-in-the-loop oversight or flexible agent orchestration. Most other solutions still rely on reactive, single-agent models triggered by explicit commands.

In contrast, optimando.ai introduces the first open-source, tab-based multi-agent framework with full local execution, real-time feedback, and complete user control. Agents operate in parallel across browser tabs, proactively enhancing workflows based on shared global goals while respecting privacy boundaries. Unlike existing approaches, this system acts as a continuous backend optimizer—augmenting user activity live, across tools and content types, in a way that remains unmatched in current implementations.

