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 for browser-based AI agent orchestration, designed to deliver context-aware, real-time optimization suggestions that proactively assist users across digital activities. Developed under the direction of Oscar Schreyer and released under the **open-source** initiative of *optimando.ai*, **a company specializing in AI automation for businesses** —the system transforms the browser into a modular orchestration environment where distributed agents enhance user intent without requiring explicit commands.

The architecture uses a lightweight browser extension to connect browser tabs to a local orchestrator desktop application. Users assign roles to tabs as either **master interfaces**—the primary point of interaction—or **helper agents**, which are exclusively implemented as browser tabs. To support repeatable and scalable workflows, the system allows entire browser sessions, including tab roles and orchestration logic, to be saved and reloaded. This eliminates the need to reconfigure sessions manually. Additionally, the orchestrator includes built-in templates for intent recognition and optimization logic, enabling the system to

proactively interpret user goals and deliver relevant suggestions without manual prompting. The AI agents in the helper tabs follow simple instructions written in plain language—no coding needed.

This is a foundational design choice: All helper agents operate within the browser and run dedicated AI models—such as ChatGPT, Claude, DeepSeek, Gemini, Mistral, Llama, Grok, or even autonomous agents like Google's Project Mariner. These models can be integrated in various ways: via APIs into your own website or app, through the providers' web interfaces, or by hosting open-source models locally on your own infrastructure.

Users set up AI agents in the app by defining simple rules like:
"Try to understand the user's intent and provide helpful information. Think a few steps ahead. Ask up to five detailed questions that might support or enhance the user's goal—but only if doing so adds real value in the current context."

This instruction is given to an initial AI agent responsible for interpreting the user's input and identifying whether deeper inquiry is justified. If the user's request is complex, strategic, or open-ended, the agent may generate an adaptive number of thoughtful, goal-oriented follow-up questions. However, for straightforward or factual queries, it may skip this step entirely or suggest only one or two relevant refinements.

When multiple follow-up questions are generated, each is routed to a separate, pre-configured AI agent running in its own helper tab. These agents work in parallel, each producing a targeted response.

*Crucially, one of the follow-up questions may be **meta-level**, such as:*
"Based on the user's current goal, what additional questions could uncover better options, reveal hidden opportunities, or help reframe the original intent for a more strategic outcome?"

*This meta-level reasoning allows the system to go beyond simply answering what was asked—it seeks to **optimize the user's intent** by surfacing smarter alternatives or future-proof decisions.*

*All answers from the helper agents are then analyzed by a **supervising agent**, which identifies patterns, knowledge gaps, and optimization opportunities. It may trigger further follow-up questions or actions—creating an intelligent, recursive feedback loop.*

*This architecture forms a dynamic **AI optimization layer** that adapts to the complexity of the task. It continuously refines, extends, and enhances the user's goals—without requiring the user to know in advance what to ask.*

In practice, a user—say, an electrician—can feed the system with documents from their electrical installation business: product manuals, safety guidelines, customer Q&As, or past project notes per email or direct upload. This data is automatically stored in a local or embedded **vector database**.

The AI agent then uses this contextual knowledge to answer questions, complete forms, or make intelligent suggestions based on user intent and even beyond. For example, if the user uploads a new project request or contract, the agent can proactively highlight missing compliance steps or suggest optimized wiring plans—based on what it has "learned" from the user's own business data.

Everything is handled within the desktop app: rules, context, and AI logic—powered by either connected APIs or local LLMs.

Each **helper agent** runs in its own browser tab and continuously analyzes the content of the active **master tab(s)**. These agents operate entirely within the browser, managed through a lightweight browser extension, and use AI models to generate real-time, intelligent suggestions that support the user's intent—whether it's increasing productivity, solving problems, or enhancing creativity.

To keep outputs organized and user-friendly, a dedicated **coordinator tab** orchestrates how executed results are displayed without delay. Instead of showing content within browser tabs, generated results are rendered directly onto the user's screen in a **structured, sectioned layout in flex boxed frames** that spans one or multiple monitors. These display sections act as dynamic slots, automatically filled with the latest agent outputs. Like this the user does not need to stare at the browser tabs while waiting for results. While the **browser-based orchestration tool** remains functional on single-screen devices, it is **not primarily designed for constrained environments**. Its full potential unfolds in **multi-screen setups and VR devices**, where the extended visual space allows for a clear, uninterrupted presentation of the parallel outputs generated by the helper agents—each displayed in **adjustable, screen-aligned panels** using flexible layout containers. This spatial separation ensures that operational core tasks and AI-generated support information can coexist without clutter or cognitive overload. Users can freely configure the number, size, and position of these display slots to match their workspace. Smooth transitions between outputs prevent visual noise or flickering. For high-speed use cases like brainstorming, users can adjust the **refresh interval** to control how often the interface updates. A **freeze button** allows locking important results in place for deeper exploration.

The interface also supports **AI-generated, context-aware action buttons**. For example, if a helper agent outputs a chart, the system may automatically create a button that lets the user switch to alternative diagram types or request deeper statistical breakdowns—providing proactive and intelligent UI elements based on the content. This browser-based

Advanced UI features like **tree-of-thought expansion** allow users to explore related ideas and branch into new directions directly from the output view. These features are optional and customizable: users or the community can define when buttons appear (e.g., on hover or in fixed positions) and what actions are triggered.

The system supports a wide range of **input channels** beyond keyboard and mouse. It can incorporate:

- **Typed text and voice commands**

- **Pointer activity and clipboard content**

- **Structured DOM data from web apps**

- **Application states and screenshots**

- **Sensor streams from external devices**, including **AR/VR interfaces** and robotic systems

All relevant input is looped through one or more master tabs, which coordinate a team of browser-based helper agents. These agents, in turn, provide intelligent, context-aware suggestions—seamlessly integrated into the user's visual workspace and controlled via the desktop application.

This architecture introduces a next-generation model of AI-powered interaction—one that is modular, transparent, user-driven, and built for high-performance multitasking across both digital and physical interfaces.

🧠 **Example Use Case: From Simple Query to Informed Decision through Multi-Agent Orchestration**

In this use case, the user interacts with the system via a **chatbot interface**, issuing a seemingly simple question:
"Can you find me a good USB stick?"

Rather than providing a single answer or a product list, the system activates a **multi-agent orchestration workflow** across several browser-based tabs. The initial user query is handled by the **master tab**, which passes it to a dedicated **helper tab** that decomposes the request into a set of optimized follow-up questions. These questions are designed not only to address the explicit need, but also to surface **relevant dimensions the user may not have considered**.

Each follow-up question is then dispatched to a **specialized AI agent** running in its own tab:

- One agent compiles a **manufacturer comparison table**, including flash chip type (SLC, TLC), hardware encryption, controller reliability, durability, shock resistance, and warranty coverage.

- Another generates **recommendations for different user types** (amateur, professional, enterprise), taking into account typical use cases, write endurance, and portability.

- A third explores **alternative storage technologies** such as **M-DISCs**, **WORM optical media**, or **enterprise-grade immutable storage solutions**, highlighting use cases where long-term durability is essential.

- A visual agent renders **comparison diagrams**, e.g. device lifespan vs. usage intensity, or chip type vs. data retention reliability.

All outputs from the agents are routed through the **coordinator tab**, which acts as an intelligent display and relevance manager. It dynamically assigns results to **display slots** on screen, grouping related outputs and prioritizing them based on contextual relevance. As the user **asks follow-up questions**, the system adapts:

- **New results are injected into available slots** or replace older, less relevant ones.

- **Highly relevant outputs remain visible longer**, ensuring continuity in cognitive focus.

- The display remains clean, organized, and semantically aligned with the user's evolving intent.

- Behind this logic is the **coordinator agent**, which does more than just route outputs—it continuously interprets the session in relation to a broader context. The system maintains a **global memory**, which may include user-provided organizational data such as company size, access requirements, data sensitivity, or strategic goals. From this, the coordinator can infer **hidden intent structures**. For example, if the user has previously indicated that the company handles critical internal documents and employs multiple staff members who need shared access, the coordinator can derive secondary goals like:
"Ensure the chosen storage medium is suitable for multi-user environments,"
"Check if the device supports encryption and access control,"

or
"Prioritize media types with long-term reliability under frequent use."
- or
"Recommend a combination of storage solutions that together cover all key advantages while offsetting individual limitations."
- These inferred subgoals work in tandem with local user queries such as *"What USB stick is fast?"* or *"Which stick lasts the longest?"*, allowing the coordinator to preserve alignment with the user's **global objectives** while guiding the session toward more robust, context-aware outcomes.

The architecture supports configurable MCP workflows, allowing users to connect LLM outputs with services such as Microsoft 365. For instance, a session summary can be converted into a structured PDF, drafted as an email, and displayed for approval and submission—automatically discarded if not submitted. Predefined templates provided by the community and Optimando.ai streamline implementation and enhance team productivity. Users can override default agent behavior with simple text instructions globally—such as disabling email draft display at session end—ensuring the system adapts to individual preferences.

⦿ Optimized Outcomes Through Collective Reasoning

What distinguishes this system is not just its ability to answer, but its ability to **reveal what the user didn't know to ask**. In this case, the user may have initially intended to purchase a USB stick to store critical personal data long-term. However, through iterative exploration guided by helper agents and managed by the coordinator tab, the system might surface the following insight:

"While USB sticks are convenient, most use consumer-grade NAND chips with limited write cycles and data retention of approximately 10 years under optimal conditions. For long-term archival of sensitive data, formats like M-DISCs or enterprise-grade WORM media offer higher durability and tamper resistance."

The system may also generate suggestions that go beyond technical specifications—such as recommending a fireproof jewel case for storing an M-DISC in a secure bank locker—if such contextually relevant additions enhance overall reliability or risk mitigation.

As a result, the user is no longer making a superficial purchasing decision, but a **strategic, informed choice**—avoiding the silent failure mode of relying on a storage medium that may degrade silently over time.

⚙️ Technical Layer: Templates and Modularity

Each agent operates based on a **system prompt template** that defines its role, behavior, and expected output format. These templates are essential to task decomposition and relevance alignment. While templates are a core architectural element, their **quality and task-fit** are critical.

To support adaptability and optimization, both **Optimando.ai** and the broader **community** will contribute to a growing library of **highly optimized, task-specific agent templates**. These templates can be plugged into any orchestration instance, allowing users to extend or adapt the system for different industries, compliance needs, or domain-specific decision support.

🔁 A Continuous Feedback Loop

Throughout the session, a **recursive feedback mechanism** ensures the system evolves with the user. If initial queries reveal gaps, contradictions, or missed opportunities, the supervising logic can generate additional follow-up questions, activate more agents, or reprioritize display slots. This **feedback loop ensures both question space and answer space are continuously refined**.

💡 **Final Outcome: Empowering Better Decisions**

By the end of the session, the user has not just received product suggestions. They've been **guided through a structured, exploratory reasoning process**, supported by domain-relevant agents, coordinated for clarity, and presented in a form that helps them act with confidence.

In the specific scenario above, the system likely **prevented a flawed decision**: using a USB stick to store critical data long-term, unaware that such media can silently degrade over time. Without intervention, that decision could have resulted in **irreversible data loss**. Through orchestration, the user was nudged toward **more durable and appropriate alternatives**—even though they had no prior awareness of their existence.

This exemplifies the system's core promise:

Turning vague user queries into structured, high-quality decisions—through intelligent, modular, and adaptive AI collaboration.

The architecture is fully open-source and designed to run locally on user-owned hardware. It includes a browser extension, a lightweight desktop orchestrator, and optional device-side input apps. There is no built-

in requirement for server-side logic. By default, all data remains on the user's system, and no external connections are made unless explicitly configured.

If users choose to enhance functionality by connecting to cloud-based language models, they remain in complete control over what data is shared. The system provides clear routing options that allow users to decide which data types are allowed to be sent to external services if the user decides to utilize external Llms. To support this even further, an optional privacy layer will be integrated that can help filter or mask typical sensitive patterns, such as names or credentials. While this layer can reduce exposure, the user always decides how much to rely on it, and can disable any external calls entirely.

For advanced users or those who prefer additional security, the entire orchestration system can be run inside a local OS in a virtual machine (VM). This adds an extra boundary of isolation and helps ensure that the AI workflow is separated from the rest of the operating system. Setting up a VM takes only minutes and requires no deep technical knowledge. Ubuntu Desktop as example is free and an excellent OS for such a VM environment. When running in a VM, users can choose to handle especially sensitive tasks—such as authentication, payments, or sensitive messaging—directly on the host system instead. This separation ensures that privacy-critical processes remain fully insulated from the orchestration environment unless explicitly bridged.

Crucially, users who want to avoid cloud services altogether can embed local language models directly into the helper tab logic. Locally hosted Llms can run inside the browser without sending any data off-device. In this case the browser simply functions as connector to the local infrastructure.

Many newer PCs now include built-in AI acceleration hardware, such as neural processing units (NPUs) or dedicated inference cores. While current browser environments offer only limited access to such accelerators, the framework is designed as a forward-looking concept that anticipates their use in future local workflows. Today, companion desktop apps or native wrappers can already utilize these components for select tasks such as inference, summarization, or intent detection, provided appropriate integration via system-level APIs (e.g., DirectML, CoreML, or ONNX). In the future, deeper browser integration with local AI hardware could enable seamless, real-time performance improvements directly in the user's browser environment. These hardware units could help facilitate lightweight, privacy-preserving automation without relying on cloud services for these critical parts, particularly when paired with local models and orchestration logic. These components can be used to speed up specific AI tasks—including local inference, redaction, summarization, or intent detection—directly on the user's device. The tab-based architecture also allows users to delegate distinct responsibilities to different tabs—for instance, using one tab to anonymize or pre-process data with a lightweight local LLM, while other tabs simultaneously leverage powerful cloud-based models, all within a controlled, user-defined workflow.

The overall design philosophy behind this framework is simple: the user stays in full control. Whether connecting cloud models, running everything locally, or blending both, the architecture adapts to individual preferences and privacy needs. It is a modular, forward-looking foundation for building intelligent, real-time workflows across devices—on the user's terms.

Unlike traditional systems that rely on a single control point, our architecture supports distributed, multi-source control. A user might speak into their phone, type on a desktop, and run a secondary application—all

at once—while helper agents receive the combined or distributed context and provide intelligent support instantly.

The system runs entirely on user-controlled devices and includes:

- A browser extension for managing helper agents,

- A desktop orchestrator app, and

- Optional input apps on smartphones, AR/VR devices, or other connected input data sources.

There is no reliance on cloud infrastructure. All logic can be executed locally, ensuring maximum data privacy, low latency, and independence from proprietary platforms. All components are open source, fostering transparency, extensibility, and long-term scalability.

This architecture represents a flexible, forward-looking foundation for real-time AI workspace automation—positioned for use in enterprise, education, science and productivity environments where data control and cross-device intelligence are strategic priorities.

The system is particularly effective in augmenting active digital workflows—whether interacting with LLMs, filling out forms, configuring automations, or managing content. It supports **real-time prompt refinement**, **tree-of-thought reasoning**, **structured brainstorming**, and **logic-driven output suggestions**. In LLM-based scenarios, the orchestrator can detect chatbot completion events and inject improved follow-up prompts automatically using DOM manipulation—enabling seamless, supervised multi-step interactions. In other use

cases, it can offer contextual next-step optimizations, alternative strategies, or compliance-aware modifications—all delivered in real time based on the user's intent and setup.

Use cases span a broad range of digital activities, including—but not limited to—automation design, form completion, business logic configuration, knowledge work, research, business communication, training, live-coaching and brainstorming. The system dynamically observes the user's intent by analyzing the visible input and output context within the active master interface. Additionally, users can define a persistent global context that reflects broader goals, project parameters, or organizational constraints—enabling the helper agents to align their suggestions even more precisely with the intended outcome. Improvements range from GDPR-compliant alternatives to optimized strategies, refined decision paths, or context-aware workflow enhancements tailored to the user's objectives.

The update interval for detecting chatbot completion, capturing screenshot or stream-based inputs, and triggering follow-up events can be precisely configured by the user.

Multi-tab orchestration

- **Multiple master tabs per session**, including support for distributed setups where multiple users, external apps, or even robotic camera systems can act as master input sources

- **Session templates** for reusable configurations

- **Autonomous and manual feedback loop triggers**: The orchestration logic allows for feedback loops between any combination of master and helper tabs. These loops can be triggered automatically based on predefined conditions, or manually by human-in-the-loop intervention. For instance, in a distributed

setup, a team of AR device operators may continuously stream contextual data into the system. Desktop-based analysts or supervisors—acting as orchestrators—can monitor this data in real time and provide direct feedback back to the AR operators. In parallel, helper tabs can exchange insights or findings among themselves based on logic rules, further enhancing the feedback cycle. This enables the creation of semi-autonomous or fully autonomous workflows, which can be toggled on or off depending on task complexity, user preference, or regulatory constraints

- **Local-only, GDPR-compliant design possible (local LLMs only)**

- A strict separation between **logic/control (master)** and **browser-based execution (helper tabs)**

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 or local open source Llms** 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 and even form filling in helper tabs; 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. Contextual signals from any application—local, remote or on premise—can be 'looped through' to the master tab, effectively integrating them into the orchestration flow.

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.

The overall architecture transforms the browser into a **distributed optimization surface** — a live environment where specialized AI agents process inputs contextually and reactively in real time. It emphasizes modularity, transparency, and cross-device orchestration, giving users full control over how intelligence is distributed and applied across their digital environment.
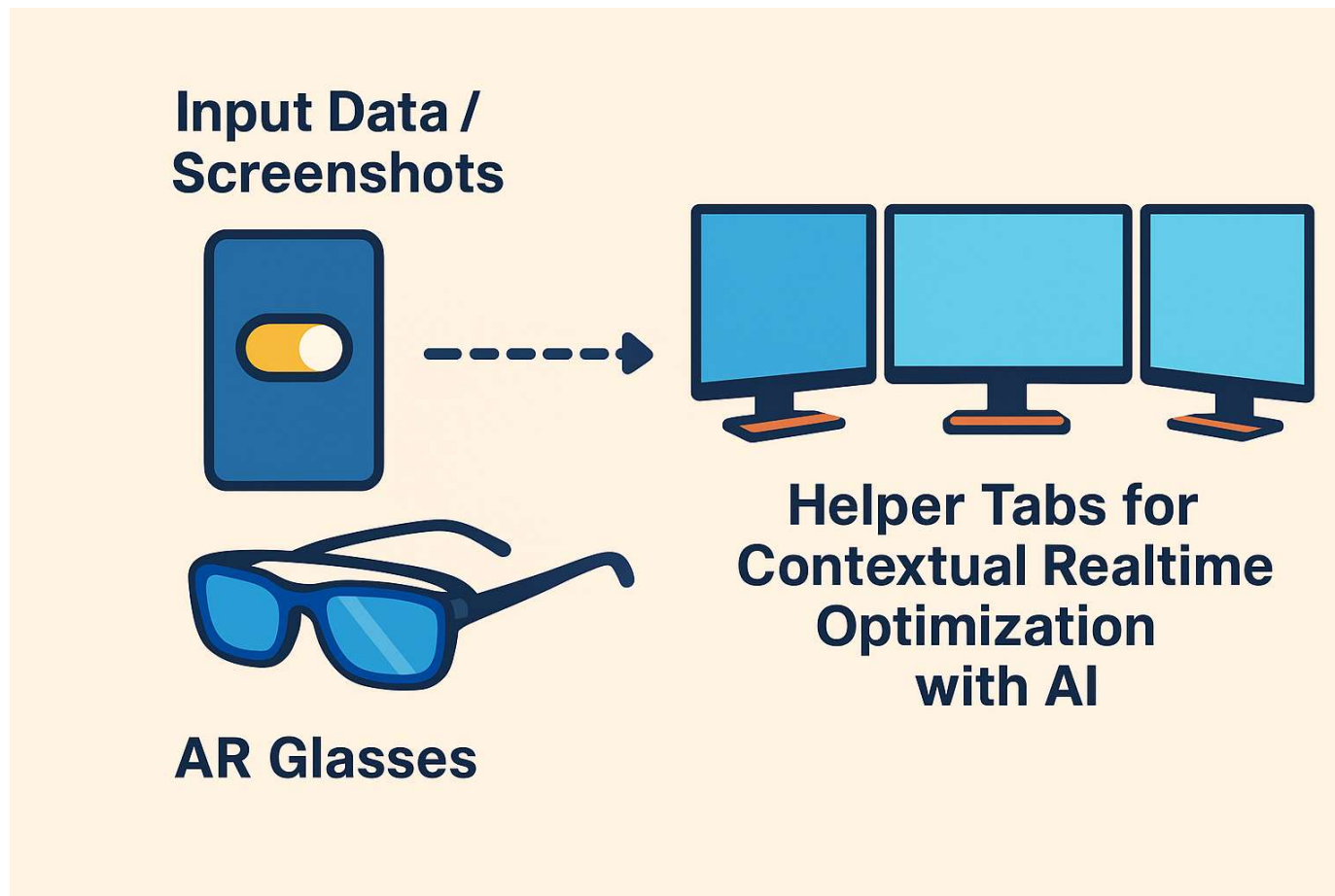
While still conceptual, the framework provides a **directionally unique and open model** for AI orchestration — enabling scalable, secure, and customizable workflows through heterogeneous agents and devices.

Contemporary AI agent tools are typically designed around a **primary agent architecture**, operating within a browser interface or cloud-based environment. For example, Google's *Project Mariner* (2025) has been described as an experimental browser assistant that allows users to interact via natural language. Based on public reports, it can autonomously navigate websites to perform actions such as purchasing tickets. More recent updates suggest that Mariner operates in a cloud-based environment, enabling concurrent task handling — though the interface remains structured around a single active agent session.

Similarly, OpenAI's *Operator* (also referred to as the "Computer-Using Agent") utilizes GPT-4o with vision to interpret and interact with web pages. Operator appears to manage multiple tasks by launching separate threads or sessions, but each instance represents a distinct agent working within its own conversational context.

Browsers themselves are beginning to integrate AI agents. Opera Neon (2025) is billed as the first "agentic browser": it embeds a native AI that can chat with the user and a separate "Browser Operator" that

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.

🧠 **Proactive DOM Manipulation via Helper Tabs with Tree-of-Thought Variant Expansion and Detached UI Controls**

The *optimando.ai* framework introduces a browser-centric, tab-based agent orchestration model in which helper agents operate in fully isolated execution environments. These agents proactively analyze structured context from the user's active session, manipulate DOM elements, and generate fully rendered outputs — all displayed outside the main interaction tab in a clean, detached format.

Unlike traditional single-path AI tools, this architecture supports **multi-path reasoning** and **alternative pre-filled results** (even before the user starts to begin with filling out fields in the master tab), activated only

when the user explicitly requests exploration via a Tree-of-Thought mechanism or other AI generated options that are displayed in the display slots.
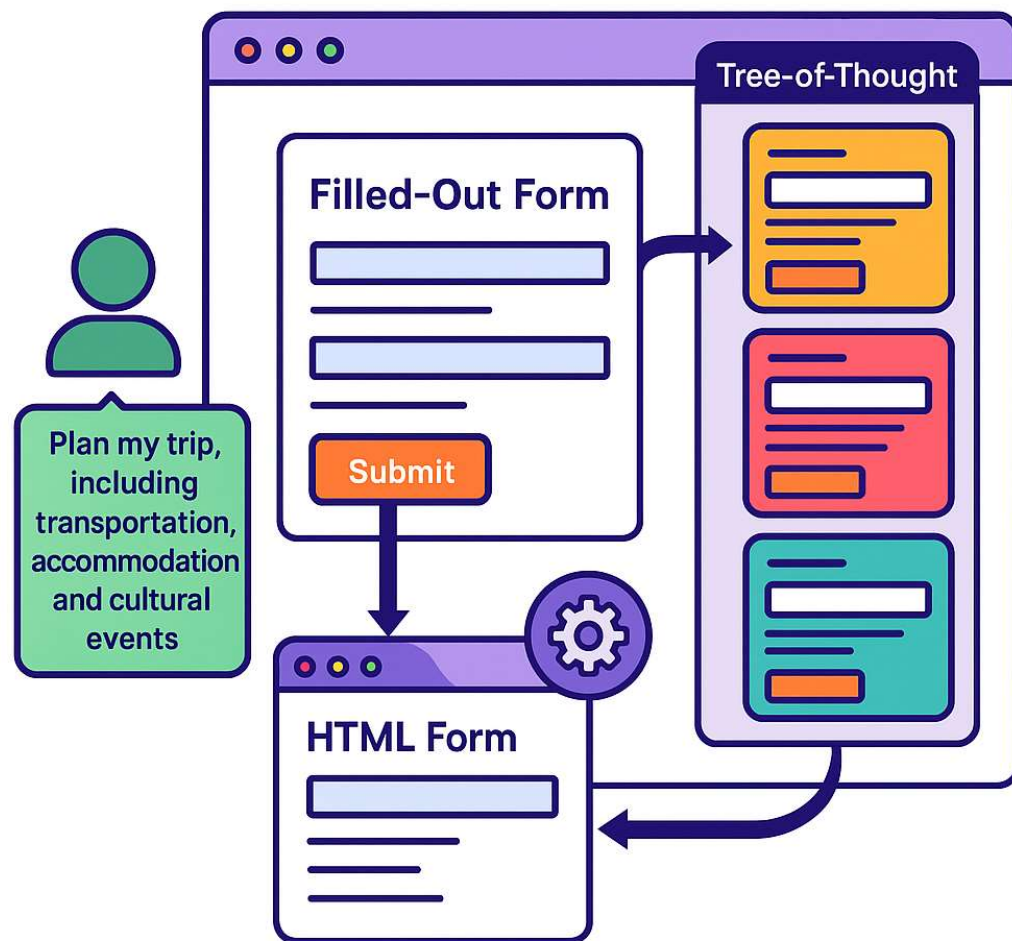
🦴 DOM Execution in Background Helper Tabs

Each helper tab can act as an autonomous agent capable of:

- Parsing DOM structures from contextual input (e.g., a form, booking flow, legal interface).

- Proactively filling out entire forms or interfaces in its own environment — without affecting the active user tab.

- Executing reasoning, retrieval, and content augmentation based on the user's prompt and global session state.

- Generating a **complete, actionable version** of the task (e.g. a filled form, contract draft, booking process) and sending it to the **orchestrator for display**.

All user-facing outputs are shown in **read-only visual clusters**, outside of the master tab, preserving the user's working context.

🧪 Generalized Example: Cross-Platform Action Planning

A user enters:
"Please plan my trip, including transportation, accommodation, and cultural events."

- A helper agent processes the instruction and autonomously interacts with multiple platforms (e.g., airline websites, hotel portals, ticketing services), while operating in separate tabs under the same authenticated session.

- It compiles the best-matching itinerary across sites, pre-fills forms where applicable, and produces a **single top suggestion**, displayed visually in a structured output slot.

- This top result includes a functional "Submit" button (non-executing by default) and, if needed, additional controls such as:

  - **Tree-of-Thought:** When clicked, this reveals **alternative variants**, each representing a different reasoning path — such as different platforms, price tiers, timing preferences, or bundled options.

    - These variants are pre-processed in their own tabs and rendered similarly.

    - The user may compare, switch between, or combine elements.

  - **Augment:** Enables real-time adaptation or enrichment of selected results.

  - **Explain:** Optionally displays why a specific variant was chosen.

Importantly, only one result is displayed by default — keeping the interface uncluttered. The Tree-of-Thought expansion is user-driven and shows pre-filled, diverse alternatives **only when actively triggered**.

📜 Additional Example: Structured Form Completion

A user begins interacting with a financial or tax-related form.

- A helper tab interprets the structure and populates it with relevant user data (pre-uploaded context data), rule-based logic, or AI-completed content.

- The best-matching version (e.g., a conservative filing strategy) is displayed first.

- If ambiguity or multiple valid interpretations are detected, a **Tree-of-Thought button** becomes available:

  - When clicked, it reveals other fully generated versions (e.g. aggressive vs. conservative deductions, business vs. private allocations).

  - Each is independently rendered, side-by-side or in cascading slots, ready for user review and approval.

🌳 What Sets This System Apart

Most existing AI assistant systems:

- Operate in a single DOM/UI context

- Most conventional AI assistants do support multiple outputs, but these are often rendered inline within the same interface, making structured comparison difficult. The suggestions are usually

presented in a linear or dropdown format, without architectural separation or agent-driven execution. This makes deeper exploration—such as Tree-of-Thought reasoning or cross-platform branching—difficult to manage or scale effectively.

- Do not support parallel reasoning paths or structured exploration

By contrast, the *optimando.ai* framework:

- **Isolates execution from interaction** — reducing risk, clutter, and interference

- **Supports cross-platform reasoning and multi-source orchestration**

- **Generates and renders variants only on demand**, using a **Tree-of-Thought button**

- Encourages structured, user-controlled decision-making, rather than opaque automation

---

✥ Architectural Benefits

- **Non-invasive assistance**: The master interface remains unchanged; all suggestions appear in dedicated display zones.

- **Session-safe integration**: Helper tabs inherit authentication from the user's active browser session.

- **Parallel reasoning at scale**: Each agent tab can target a different platform or strategy — executed in true parallelism.

- **Human-in-the-loop control**: Results are passive unless approved; the user always decides what to apply or explore further by default.

🔐 Privacy by Architecture: Local Control Through Tab-Based Orchestration

Modern AI automation presents a paradox: its usefulness grows with access to user context — but so do the privacy risks. *optimando.ai is designed to support users in protecting their privacy through a layered, modular system architecture. While it cannot prevent all risks — especially when users voluntarily share personal data with external services — its structure enables masked automation, local execution, and transparent control over agent behavior.*

As an open-source platform without vendor lock-in, optimando.ai invites continuous improvement and innovation. Its flexible design allows the community to build on a privacy-aware foundation that evolves with real-world needs.

At the heart of the system is a browser-centric tab orchestration model, which allows users to automate workflows while preserving control over how, when, and what kind of data is processed.

📦 Layered System Components for Structural Privacy

This is not abstract or theoretical privacy; it is embedded directly in the system's architecture and enforced through clearly separated roles and control layers.

Component	Privacy Role
	The user's primary interaction layer — which may include but is not limited to browsing, shopping portals, SaaS tools, embedded apps, or even external visual inputs (e.g., camera feeds, video streams, robots). This is the starting point for all activity.
Master Tab	Because the Master Tab handles real-time input directly from the user and external sources, *optimando.ai* cannot intercept or mask visual or textual content before it reaches cloud-connected systems. Users must be aware that any personal data or visual context shared here can be exposed, especially if routed to external AI services.
	optimando.ai provides tooling to structure, automate, and augment downstream tasks — but *cannot enforce privacy at the source input level. The user remains in control.*
Helper Tabs	Sandboxed environments where AI agents operate. These tabs are designed to receive masked, preprocessed, or synthetic data, especially when handling structured automation tasks like form filling or contextual suggestion.
	However, the degree of masking depends on user configuration and workflow context. *optimando.ai* provides mechanisms to prevent the exposure of raw personal data, but cannot enforce this universally.

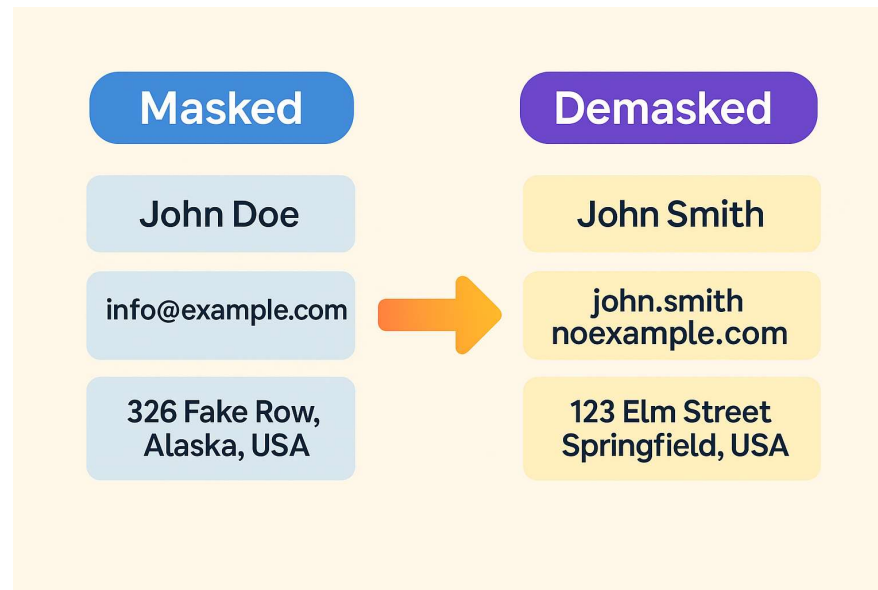| Component	Privacy Role
	It is the user's responsibility to ensure that privacy settings are correctly applied and that no sensitive inputs are embedded into prompts or data sent to cloud-based models without proper masking.
Coordinator Tab	Routes logic, manages masking/demasking workflows, and controls agent output. It ensures context flow is constrained and reversible locally.
Display Slots	Passive render areas. They display results from helper agents but do not trigger outbound data flow. If the user chooses to submit a form, it's sent directly to the target website, not to any AI system.

This orchestration strategy allows privacy protection at the execution level, not just via policy.

🛡 What's Protected: Contextual Data Masking

The system applies local masking and demasking to sensitive fields before sending them to helper agents. Supported transformations include:

- Names → pseudonyms (e.g., "John Doe")

- Addresses → synthetic or shifted variants

- Dates → randomized or offset

- **Numbers → obfuscated mathematically (e.g., −30% for income)**

Masked	Demasked
John Doe	John Smith
info@example.com	john.smith noexample.com
326 Fake Row, Alaska, USA	123 Elm Street Springfield, USA

Example: A user opens a tax declaration form, and *optimando.ai* activates one or more helper agents to simulate pre-filled version(s). To protect privacy, the system masks sensitive data — such as reducing income values or substituting names and addresses — before passing the input to the AI agent. On premise the masked data will be demasked again so that the user sees the pre-filled forms with real data.

- In many automation scenarios, these masked values are sufficient for meaningful reasoning: the AI can still understand the form structure, recommend deduction strategies, or identify missing fields without relying on exact personal data. The user later sees a demasked, accurate version — rendered locally in a reviewable format.
- The overall design of *optimando.ai* aims to support **on-premise masking, demasking, and Tree-of-Thought-based confusion**, where multiple form variants are generated in parallel and only the system internally knows which one reflects the user's real data or decision. This makes external profiling extremely difficult.
- Additionally, the framework is intended to highlight potential **augmentation risks** (e.g., when AI decisions rely on data that cannot be effectively masked) before agent execution.
- It is important to note that **not all features will be fully functional from the beginning**. *optimando.ai* is an open-source, evolving architecture — designed to grow with community input and to gradually deliver **realistic, technically feasible privacy protections** in AI-assisted workflows.

🌲 Tree-of-Thought Expansion & Intent Obfuscation

Thanks to its distributed helper-tab design, *optimando.ai* enables a novel privacy feature: intent obfuscation through variant generation.

- **Multiple helper agents generate alternative form completions (e.g., booking dates, tax strategies, application paths).**

- **Only some are "real" — but the AI doesn't know which. The user sees only real results.**

- **The others act as plausible decoys and are processed in the helper tabs.**

This makes it difficult for even a remote AI system to infer true user preferences or intent — especially in planning and decision-heavy workflows.

The concept is simple: the more plausible paths exist, the harder it is to profile the user. This concept
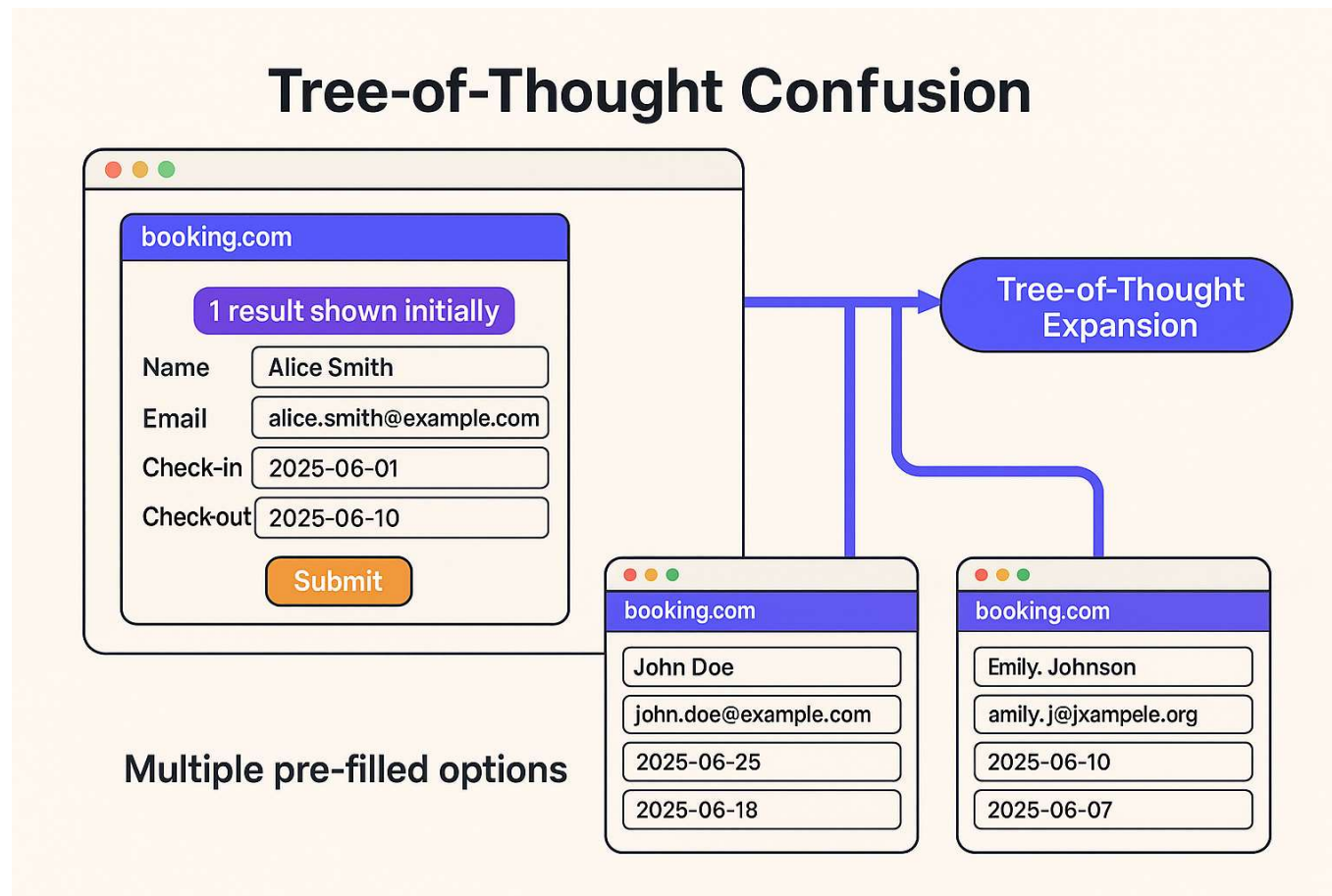
Conceptual Feature: Tree-of-Thought Confusion and Profiling Risk Mitigation for master tabs

A privacy-first concept currently in internal design is the Tree-of-Thought Confusion mechanism, paired with real-time profiling risk detection and dynamic masking/demasking overlays in the Master Tab. This approach is designed to protect users against unintended profiling, especially when interacting with LLM-based systems in sensitive contexts such as health, legal, or financial domains.

Example Use Case

Consider a user entering the prompt:

"Please find a list of oncologists near Berlin for my ongoing cancer treatment."

Although the request may appear routine, it contains highly sensitive health information. If such input were to be processed by an external LLM or cloud service without obfuscation, it could contribute to a personal profile — potentially exposing the user to discrimination (e.g., by insurers or employers). Most users are unaware of such risks, especially given the normalized tracking behavior in consumer platforms like search engines or video portals.

How It Works Today

In its current concept form, the system does not fully rely on AI to detect sensitive content. Instead, it uses lightweight, on-premise techniques, including a small local Llm like mistral 7b:

- Keyword and pattern matching (e.g. "oncologist," "HIV," "diagnosis," "disability", "Doctor")

- Prompt structure heuristics (e.g. first-person health-related statements)

- DOM context awareness (e.g. recognizing field types like medical_input or financial_reason)

When a high-risk context is detected:

- The system delays prompt execution through DOM interception,

- Generates multiple prompt derivatives with decoy health topics,

- Sends those prompts to various helper tabs to confuse external systems about the user's real intent.

- All decoy prompts are processed in background helper tabs.
  Only the real result is routed to the display slot on screen, as the system knows which output matches the true intent.
  While all prompts are technically visible in the helper tabs, only the correct answer is shown in the main interface.

- Applies additional masking/demasking locally, making reverse profiling extremely difficult.

This approach is technically feasible and highly valuable but also complex to implement robustly, especially under real-world DOM variability and multi-agent tab environments. It is therefore not included in the early versions of the platform but targeted for future development.

Future Evolution with Local AI

As local hardware continues to evolve, it is expected that small LLMs integrated directly into consumer PCs and laptops will become standard. These on-device models (e.g. 3–7B parameters) will eventually be capable of:

- Real-time sensitivity detection across diverse inputs and domains,

- Context-aware obfuscation, adapting masking strategies intelligently,

- Improved Tree-of-Thought Confusion logic, simulating plausible decoy intent patterns with high accuracy.

Such models would run entirely on the user's machine, enabling advanced protection without compromising privacy or requiring a cloud connection.

Design Rationale

This concept addresses a major barrier for privacy-conscious users who avoid AI agent systems altogether due to unresolved profiling risks. By embedding privacy-sensitive reasoning directly into

the orchestration layer — first via rules and heuristics, and later via embedded LLMs — the platform aims to build long-term trust and give users real agency over how and when sensitive data is processed.

🔒 **Local-Only Form Submission**

When an agent completes a form, it appears in a Display Slot. If the user chooses to submit:

- **The submission goes directly to the target site (e.g., a booking or government portal)**

- **No AI model or external agent sees the unmasked data**

- **Execution happens via the user's own session and browser state**

This makes the entire interaction local, transparent, and user-driven.

⚠ **What Can't Be Protected**

This system is powerful, but it's not magic.

- **If a user types their real name, address, or salary into a public chatbot — that data is exposed.**

- **If a task requires semantic accuracy (e.g., legal document interpretation), masking may break the logic.**

optimando.ai **does not claim to prevent all leakage. It offers technical control where feasible, and leaves informed decisions to the user.**

🧠 The User Is Always in Control

Privacy in *optimando.ai* isn't based on trust — it's based on architecture. But it also assumes that:

- The user understands what happens where

- The user decides what is masked, unmasked, or revealed

The system never processes or acts autonomously on private data without explicit user permission. All integrations, logic flows, and connected services are defined by the user. The user remains fully in control of which tools are used, how data is routed, and when execution is allowed. This is human-in-the-loop AI orchestration — not blind automation.

☑ Summary: Real Privacy, Built In

optimando.ai enables advanced AI automation — form completion, variant suggestion, workflow support — without exposing user identity, sensitive values, or decision logic to remote models. Through tab orchestration, masking, and local submission, the system offers:

- Agent-level PII protection

- Tree-of-Thought obfuscation

- Local demasking and rendering

- Full user control at every step

What makes this system stand out is not just what it does — but what it lets the user choose not to do.

Conceptual Feature: On-Premise Augmentation Overlay for Masked Reasoning

As part of future development, we propose an experimental feature designed to improve the interpretability and safety of decision-making under data masking: a Master UI overlay that dynamically highlights critical input fields and allows users to simulate randomized variations of masked data locally.

This concept—referred to internally as the Augmentation Overlay—would provide a user-facing mechanism to explore how masked or obfuscated data influences the system's reasoning path. The feature is intended only for on-premise execution and is designed to ensure that no sensitive data is ever transmitted or logged externally. The overlay itself could be toggled on or off. After all we aim for a fully autonomously real-time user intent and prediction loop with forward-thinking real-time support.

Key Ideas Behind the Overlay Concept

- **Dynamic Field Highlighting**
  The Master UI would automatically detect which input fields are involved in the current masking/demasking logic and visually augment them using a transparent overlay. Fields may be color-coded or animated to indicate their importance in the current reasoning context.

- **Simulated Input Randomization**
  Users can press a button to generate plausible random values for any field involved in the masking logic. This enables counterfactual simulations: "What would the system conclude if these values were different?"

- **Human-in-the-Loop Reasoning Validation**
  By observing how simulated changes affect the output, users can monitor whether the reasoning engine reacts in unexpected or overly sensitive ways. This strengthens transparency and helps identify fragile dependencies in automated decision logic.

- **Field Sensitivity Feedback (optional extension)**
  Future iterations of the concept could introduce a sensitivity score per field, highlighting which data points have the greatest impact on logic flow when masked or altered.

Privacy and Security Implications

Importantly, all logic and transformations would occur exclusively on the user's machine. The feature is meant to aid local inspection and debugging. As such, it aligns with strict privacy requirements (e.g., GDPR, HIPAA, and industry-specific compliance standards).

Future Hardware Considerations

As edge computing and AI-capable hardware continue to evolve, it is foreseeable that on-device LLMs—running on secure modules or embedded AI accelerators—could take over the localized reasoning logic for such privacy protection layers. These compact models, optimized for low latency and high privacy, would enable real-time counterfactual testing and field sensitivity analysis without requiring a cloud backend. However, the decision-making of what data could be critical can also be done in the cloud as that's no sensitive data, if that makes any sense. Such developments would further decentralize AI decision auditing and make this overlay concept viable for regulated or infrastructure-constrained environments.

Development Status

This feature is currently a conceptual proposal and will not be part of the initial release. It represents an advanced step toward interactive AI debugging, and its feasibility, usability, and performance implications must be carefully evaluated before full implementation.

Nonetheless, the Augmentation Overlay aligns with the broader vision of giving end users greater control, visibility, and confidence when interacting with masked data and semi-autonomous agent systems.

✳ Use a Dedicated Virtual Machine (VM) or Isolated Workspace

It is recommended to run the orchestration environment (including Master Tab, Coordinator, and agents) inside a dedicated virtual machine or isolated operating environment.

This provides two core benefits:

1. Technical Isolation – Prevents cross-contamination of session data, cookies, or clipboard content between personal activity and AI workflows.

2. Contextual Awareness – Using a VM helps users mentally distinguish between "AI mode" and regular activity, reducing the risk of unintentionally exposing sensitive data.The separation acts as a continuous reminder that interactions may be processed, embedded, or analyzed — and should be treated accordingly.

🌐 Use a Separate Browser Profile or Session

For users not using a full VM, it is advised to run *optimando.ai* in a dedicated browser profile or container session. This limits access to personal cookies, autofill data, and login states that may otherwise be unintentionally exposed to AI agents or helper tabs.

❇️ Masking Defaults and Manual Overrides

Users should:

- Review and configure default masking rules (names, addresses, numbers, etc.) before running automation tasks.

- Use manual overrides only when necessary — e.g., for document reasoning tasks that require semantic fidelity.

- Understand that masking only applies within agent-controlled flows and does not affect direct input into third-party AI systems (e.g., public chatbots).

🚫 Avoid Manual PII Entry in Master Tabs Connected to AI-Services

The Master Tab is the entry point to many workflows. While it is architecturally isolated from helper agents, it may still be connected to external systems (e.g., websites, LLM chat interfaces).

Users are responsible for avoiding direct entry of personal, financial, or sensitive data into services where masking cannot apply.

📲 Prefer Local or Air-Gapped Modes for High-Sensitivity Tasks

When working with proprietary, legal, or highly sensitive data, users should:

- Prefer offline LLMs or local inference engines

- Disable or restrict outbound AI service calls entirely

- Inspect agent logic manually before execution

optimando.ai is fully open source and supports air-gapped use cases — making it suitable for deployment in secure environments.

The orchestration system's source code is published under the **GNU Affero General Public License v3 (AGPLv3)**, reflecting a strong commitment to **open-source principles** and ensuring derivative works remain open, particularly in network environments. The conceptual framework and accompanying documentation described in this paper are licensed under the **Creative Commons Attribution-ShareAlike 4.0 International License (CC BY-SA 4.0)**. This dual licensing model for the software, along with the **CC BY-SA 4.0** for conceptual works, ensures both robust open-source integrity and safeguards for attribution and openness across all project artifacts. For commercial, closed-source, or enterprise environments requiring alternative terms (including specific UI attribution mandates), a separate commercial license is available.

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 helper tab outputs even from possible other remotely interconnected helper tabs or input sources as suggestions or context inserts in real time. This creates an optimization loop: the user steers the main task, customizable helper tabs continuously augment it automatically, and the user approves or refines the results. The human stays "in the loop" at every step, aligning with best practices in trustworthy AI.

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:

Novelty of optimando.ai's Approach

Given the landscape above, **optimando.ai's** combination of features **does appear to be novel and unmatched**. In particular, no known system offers *all* of the following in one package:

- **Fully local, browser-native multi-agent orchestration:** Many systems run in the cloud or require server components. Those that are local (browser extensions like Nanobrowser or RPA tools) don't typically orchestrate multiple autonomous agents across several browser tabs without user direction. Optimando's design of a local master tab coordinating slave tabs for different subtasks is unique.

- **By integrating directly into the browser landscape—the primary interface for digital activity worldwide—optimando.ai enables real-time optimization or intelligent, context-aware, goal-driven optimization suggestions at scale, putting AI orchestration into the hands of every user without relying on proprietary platforms, cloud dependencies, or specialized infrastructure**

- **Autonomous, proactive assistance:** Most current solutions are *reactive*. They await user queries or commands. A system that observes the user's context and proactively generates suggestions or carries out optimizations (e.g. automatically augments your task flow across multiple sites) is not mainstream yet. Yutori's "Scouts" come close conceptually (monitoring in the background)github.com, but those operate on specific user-defined goals (like watching a particular site or alert type) **rather than**

generally optimizing any ongoing browsing activity. An *AI that feels like a colleague actively helping unprompted* is largely aspirational right now.

- **Multi-agent parallelism in a user-facing application:** While research and some closed prototypes leverage multiple agents in tandem, typical user-facing AI assistants are single-agent. Optimando.ai's vision of parallel agents (each potentially with specialized roles or focusing on different tabs) coordinating in real time to help the **user is cutting-edge**. We see early signs of this in Opera Neon's ability to multitask and in Nanobrowser's planner/navigator duo, but these are either constrained or not autonomous. **No product fully utilizes a swarm of browser-based agents to continuously adapt to what the user is doing.**

- **Context-aware cross-tab optimization:** This implies the system maintains a high-level understanding of the user's objectives across multiple browser tabs or tasks. None of the surveyed tools truly does this. For instance, if a user is doing research with several tabs, current AI assistants might summarize one tab at a time when asked, but they won't *on their own* consolidate information from all tabs or reorganize them for the user's benefit. **Optimando.ai** aiming to provide "**real-time, context-aware optimization**" suggests it would do exactly that – **something quite novel**.

In conclusion, the core architecture of *optimando.ai* – a local master–slave tab framework enabling an autonomous multi-agent assistant system – **is novel**. Existing systems offer pieces of the puzzle (cloud-based autonomy, local extensions, multi-tab tools, etc.), but none delivers the same integrated experience. Therefore, **optimando.ai's** implementation would represent a distinct advancement in browser AI orchestration and autonomy. Its closest peers (Google's Mariner, OpenAI's Operator, Opera's Neon, academic

Orca, and various extensions) each lack at least one crucial element (be it full local execution, open-source, proactivity, multi-agent parallelism, or deep context integration). As such, optimando.ai's concept stands out as unmatched in combining all these features into one system, **marking a potentially significant innovation in the AI browser assistant space.**

References: The analysis above references key details from current systems, including Google DeepMind's Project Mariner[techcrunch.comtechcrunch.com](#), OpenAI's Operator[techcrunch.comtechcrunch.com](#), UCSD's Orca browser research[arxiv.orgarxiv.org](#), the Nanobrowser open-source project[github.comgithub.com](#), Opera's AI initiatives (Aria and Neon)[press.opera.compress.opera.com](#), and curated lists of web AI agents and automation tools[github.comgithub.com](#). Each of these informs the feature comparison and underscores the novelty of optimando.ai's approach.

Key innovations of the optimando.ai framework include:

- Master–slave tab architecture, where each browser tab runs a dedicated AI agent with a defined role in the workspace.

- Real-time, context-aware optimization, where agents interpret and enhance user workflows without explicit commands.

- Parallel agent execution across tabs, allowing intelligent coordination of tasks like data entry, content drafting, monitoring, and summarization.

- Global goal propagation, enabling all agents to align with high-level objectives defined by the user or inferred from context.

- Local-only architecture, ensuring full privacy, transparency, and auditability—no hidden cloud inference or data leakage. The user has full control over the data flow.

Unlike Mariner or Orca, optimando.ai does not wait for the user to act. It acts with the user, continuously enhancing workflows as they unfold—across tools, content types, and digital services. It is not a helper or assistant. It is a real-time orchestration layer for the modern web workspace.

To our knowledge, no existing system—academic or commercial—combines autonomous multi-agent orchestration, tab-based modularity, proactive real-time augmentation, and privacy-first, local execution. This positions optimando.ai as a breakthrough platform in browser-native AI automation.

🔥 From Real-Time Gaming to Real-Time Intelligence: A Paradigm Shift

Modern gaming has become a proving ground for real-time computing performance. Today's top-tier systems deliver:

- **🎮 144–360 FPS forward rendering,**

- **⏱ Sub-10ms input-to-photon latency,**

- **🧠 DLSS/Frame Generation by AI,**

- **and instant asset streaming over hybrid cloud-edge setups.**

🧠 High-Fidelity, Real-time Optimization Towards Predicted or Defined Goals

These same principles—low-latency responsiveness, forward-thinking, dynamic rendering, and distributed compute—are now crossing into productivity and automation. Breakthroughs on multiple levels will make unimaginable things possible within the next decade and this conceptual browser orchestration framework puts this power into the hands of everybody with a pc and internet access. After all advanced AI-driven fast-pace gaming compute is similar to real-time data generation.

Imagine a knowledge worker's future desktop setup:

🖥️ **Screen 1:** A browser helper tab hooked to an LLM refines every question and interaction you write—augmenting your thinking through prompt optimization and chain-of-thought amplification.

📊 **Screen 2:** Another tab visualizes live data from an internal MCP (Multi-Channel Processing) server, rendering interactive, high-frequency charts in real time using forward-rendering browser tech via WebGL or WebGPU.

🎬 **Screen 3:** A helper agent watches your actions and assembles a narrated video tutorial using generative AI—documenting decisions, insights, and process flows as you work.

🧬 Every helper tab runs in a secure, browser-isolated environment, each functioning as a dedicated AI agent. The orchestrator ensures timing, logic control, and agent-to-agent feedback loops, all under user supervision.

🛠 The Technical Foundation

Unlike traditional agent systems that rely on centralized cloud logic or bespoke SDKs, the optimando.ai framework is built on:

- Browser-native orchestration using tabs, not containers

- DOM-level prompt injection and readback, per desktop/mobile app and browser extension

- User-defined session templates and context-aware routing logic

- Customizable update intervals (DOM completion, screenshot loop, stream window)

- Autonomous or manual feedback triggers, including peer-to-peer helper tab interactions

- Full MCP server compatibility via orchestrator logic via helper tab (local/remote event listeners)

- Hybrid LLM handling, where each helper tab can run:

  - Local models (e.g., Mistral, LLaMA, Phi-3)

  - Cloud models (e.g., GPT-4, Claude, Gemini, Deepseek, Mistral)

  - Or even autonomous agents (e.g., OpenDevin, Project Mariner)

- Security by Design through browser sandboxing, session isolation, and non-invasive architecture

-

- The browser, long seen as a passive interface, is now the orchestrated runtime layer.

• Security by Design: The system leverages native browser sandboxing, session isolation, and a non-invasive architecture (no root access, no background daemons), reducing attack surfaces and simplifying compliance.

• Modern autonomous agents—such as OpenDevin, Google's Project Mariner, or Baidu's Ernie Bot Agent—highlight the global trend toward AI-driven process delegation. However, many existing solutions remain closed, platform-bound, or require deep system integration. Optimando.ai takes a more flexible route: its browser-based helper tab concept allows users to integrate AI agents and automation tools—including LLMs, n8n, Zapier, Make, or other cloud/local services—without leaving the familiar browser environment.

• This architecture enables real-time orchestration of AI workflows and brainstorming sessions across browser tabs, supporting both cloud-based APIs and fully local execution. It offers a modular and scalable framework that allows organizations to incrementally adopt AI-driven automation—without lock-in, without compromising data ownership, and with minimal infrastructure requirements. The result is a powerful, interoperable AI workspace that aligns with existing digital behavior while opening the door to highly personalized and responsive task automation.

🌐 The Browser as a Universal AI Gateway

Why the browser? It is universally available, cross-platform, and runs on every device

- It has evolved with WebGPU, Service Workers, Security Sandboxing, and full user-level isolation

- It allows interaction with any digital tool or AI system that exposes a UI

- It is where 98%+ of digital work happens—from cloud IDEs to enterprise dashboards

optimando.ai leverages this to orchestrate entire multi-agent workflows from within the browser, controlled by a single orchestrator app on your device and a browser addon. No need for server-side logic—only tabs. For users seeking maximum privacy and control, the orchestration tool can be installed on a bootable, encrypted SSD preconfigured with a hardened Linux distribution such as Ubuntu. This setup allows the entire orchestration environment—including the browser-based agent system and supporting components—to run in an isolated, portable, and tamper-resistant workspace.

To simplify deployment, Optimando.ai will offer a ready-to-use secure setup, which includes full disk encryption, pre-installed orchestration software, and optional integration of local language models (LLMs) for fully offline workflows. This approach is ideal for professionals, researchers, or organizations that require both AI automation and strict data sovereignty.

🔐 Secure Deployment via Bootable Encrypted SSDs

To support privacy-focused and offline use cases, the orchestration framework can be deployed on a bootable SSD with full-disk encryption, running a desktop-capable Linux distribution such as Ubuntu 22.04 LTS Desktop Edition. This configuration allows the entire orchestration system—including the browser-

based interface, coordination logic, and helper agents—to run in a graphical, self-contained, and tamper-resistant environment.

The inclusion of a full desktop environment is essential, as it provides the graphical interface needed for interacting with browser tabs, multi-agent outputs, and visual workflows—similar to how a typical end-user system operates (e.g., on Windows or macOS).

For maximum security, the entire device should be encrypted using technologies like LUKS full-disk encryption, ensuring that no part of the disk remains exposed to boot-time malware or unauthorized data access. The system image can also be cryptographically hashed to enable post-installation integrity checks and reproducible builds. Additional hardening measures such as secure boot and optional air-gapped operation may be applied depending on the threat model.

This deployment strategy is particularly useful in environments with strict data governance policies, limited or no internet access, or where offline, local AI processing is required.

🏳 A Timestamped Innovation. First Public Release of Its Kind.
The conceptual design and technical outline of this system were first made publicly accessible in 2025 and cryptographically timestamped using OpenTimestamps on the Bitcoin blockchain. This verifiable proof-of-publication ensures authorship precedence and protects against future claims of originality.

📌 To the best of our knowledge, this was the first publicly released open-source orchestration framework enabling browser-central, tab-based AI agents to coordinate, optimize, and suggest real-time strategies—across devices, sessions, and users. Unlike traditional automation tools that react only to explicit user input, this system is built around proactive, forward-thinking, continuous monitoring and intelligent feedback loops. Helper tabs autonomously observe context and suggest optimizations without requiring manual triggers—delivering a dynamic, adaptive AI experience across the user's digital workspace.

Concept Timestamped on Bitcoin

A TIMESTAMPED INNOVATION. FIRST PUBLIC RELEASE OF ITS KIND.

OPTIMANDO.AI

A BROWSER-BASED, TAB-BASED AI ORCHESTRATION SYSTEM ENABLING AGENTS TO COORDINATE, OPTIMIZE, AND SUGGEST REAL-TIME STRATEGIES ACROSS DEVICES, SESSIONS, AND USERS. PUBLICLY RELEASED IN 2025 AND CRYPTOGRAPHICALLY TIMESTAMPED USING OPENTIMESTAMPS ON BITCOIN BLOCKCHAIN.