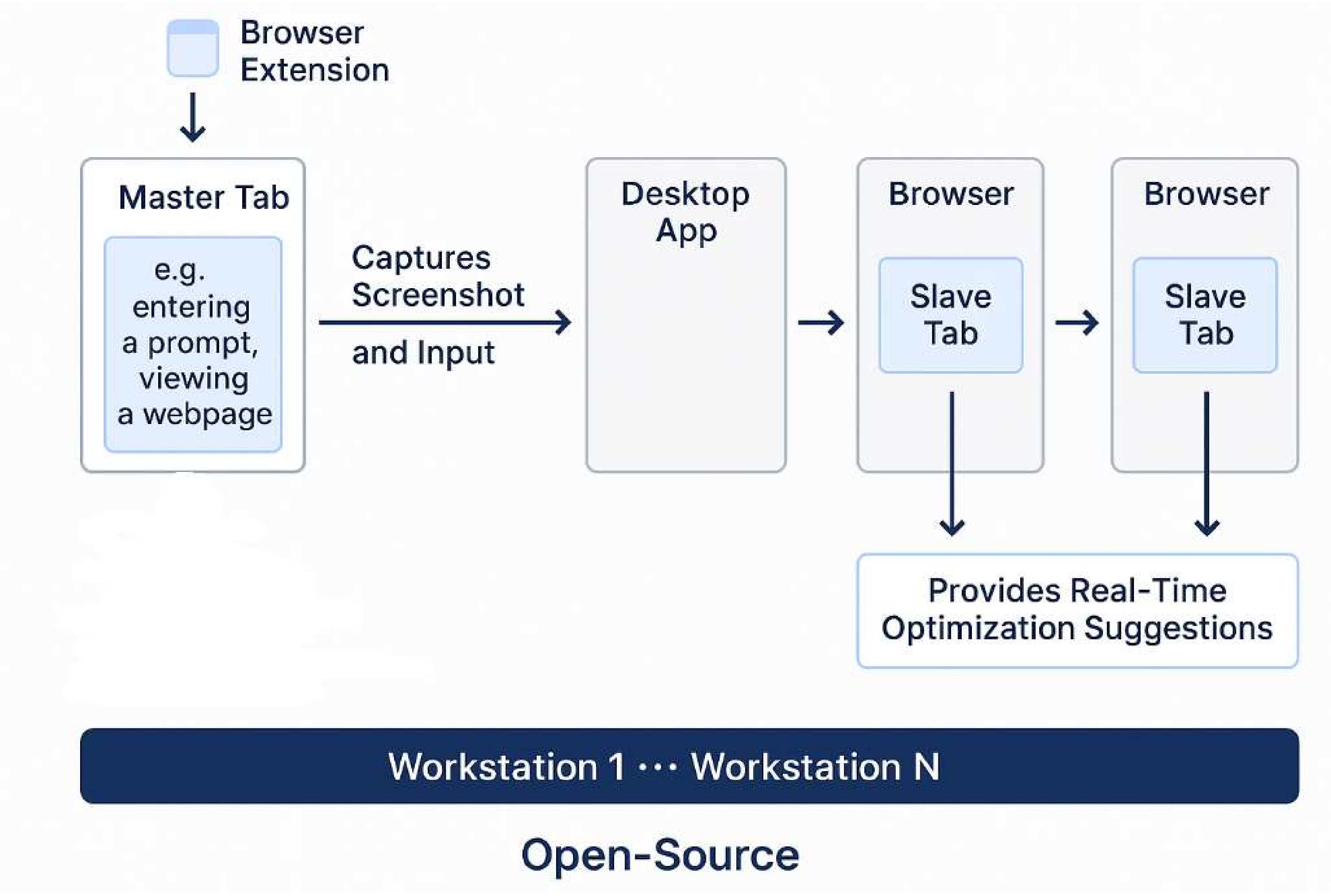
OpenGiraffe – Open-Source Browser-Based AI Agent Orchestration Framework


Browser Extension
Master Tab
e.g. entering a prompt, viewing a webpage
Captures Screenshot
and Input
Desktop App
Browser
Slave Tab
Browser
Slave Tab
Provides Real-Time Optimization Suggestions
Workstation 1 ··· Workstation N
Open-Source

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 with the name **OpenGiraffe**, 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 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 systems like Google’s Project Mariner. These models can be integrated in various ways: via APIs into a custom website (enabling community-driven templates), through the providers’ web interfaces, or by hosting open-source models locally—typically exposed through a browser-based UI as well.

In the default OpenGiraffe architecture, each AI helper and AI coordinator agent is hosted in its own browser tab. This setup is essential when working with web-based LLM chatbots, where the model interface is embedded in the frontend of a website and cannot be accessed via direct API calls. In such cases, the browser orchestrator must manage real tabs to maintain context and simulate user interaction.

However, when LLMs are integrated via APIs—whether from cloud providers or locally hosted models—it becomes technically feasible to consolidate all agent logic within a single custom-built website, running in a single browser tab. This orchestration page is hosted locally on-premise and defines each AI helper and coordinator agent using structured HTML snippets with embedded metadata (e.g., unique IDs, roles, endpoints, reasoning logic, and parameters). The browser extension then reads and executes the orchestration logic directly from this unified interface.

The unified page supports multiple profiles and modular AI agent configurations, each of which can be independently toggled—enabling flexible activation of specific workflows or toolchains. Hosting the orchestration layer on-premise not only enhances privacy and data sovereignty, but also enables safe, community-driven sharing of reusable agent templates without exposing backend infrastructure.

Functionality remains consistent: master input tab context is preserved, and agent outputs are routed to display slots—while the traditional "one tab per agent" abstraction becomes optional in API-driven environments.

🔁 Adaptive Agent Architecture for Context-Aware Reasoning

Users configure AI agents in the system by defining behavior rules that guide how user input should be processed. A typical instruction might be:

“Interpret the user’s intent not only from the immediate input, but also from broader context, historical memory, and ongoing interaction patterns. Distinguish between short-term goals and long-term objectives. If helpful, generate up to five targeted follow-up questions or suggestions that either: (1) directly support the short-term goal, or (2) uncover smarter alternatives, missing steps, or overlooked opportunities that better align with the user’s long-term intent—even if these haven’t been explicitly requested.”

This high-level rule is executed by a chain of modular agents that work together to analyze, extend, and optimize user input dynamically.

🧭 Step 1: Input Coordination and Intent Expansion

All user input is first processed by an Input Coordinator Agent, which evaluates the complexity and intent of the request. It distinguishes between:

- Simple, factual queries → Direct response or light refinement
- Complex, strategic, or ambiguous inputs → Deeper inquiry and intent optimization

In the latter case, the agent generates follow-up prompts designed to clarify goals or surface smarter alternatives. These may include meta-level questions, such as:

“What additional questions could help uncover better options, reveal blind spots, or help the user reframe their original goal for a more robust outcome?”

“Could the user be aiming for an outcome they haven’t articulated, possibly due to lack of technical vocabulary or system awareness?”

Unlike standard prompts, meta-level questions do not produce direct answers. Instead, they return new follow-up questions as output. This behavior triggers an internal feedback loop:

1. The meta-agent receives the original context and task framing.
2. It generates a set of secondary follow-up questions; each aimed at improving strategic alignment or surfacing hidden insights.

3. These secondary questions are routed to new Helper Agents, just like primary ones.
4. Only once those agents return their responses does the system proceed to synthesis and output.

This recursive mechanism allows the system to reason not only about *what* the user asked, but *whether they're asking the right questions*—turning the orchestration into a continuously optimizing cycle.

🧠 Step 2: Distributed Parallel Reasoning via Helper Agents

Each follow-up question is then delegated to a specialized Helper Agent, which processes it independently. These agents work in parallel, using tailored logic and data access scopes.

There are two execution modes:

1. Browser-Tab Mode (**OpenGiraffe:** Default Web-Based Architecture)

Each Helper Agent is launched in a dedicated browser tab, managed by a browser-based orchestrator. This setup is essential when working with web-based LLM frontends (e.g., ChatGPT, Claude) where API access is unavailable. The orchestrator simulates user interaction across tabs to maintain context and autonomy for each agent.

2. API-Based Mode (Custom Website Integration)

When LLMs are accessible via API, all Helper Agents can run inside a single custom-built web interface. Each agent is defined as a modular component (e.g., HTML + metadata) with a unique ID. The browser extension routes queries directly to the appropriate module within the page—allowing agent logic to execute concurrently without multiple browser tabs.

Regardless of the mode, Helper Agents may utilize:

- Structured memory from a local or embedded vector database
- Context-specific data filters (e.g., safety documents only)
- Agent-specific prompt engineering or tool access

This modular parallelism enables domain-specific specialization, such as pricing analysis, compliance verification, or user behavior prediction.

📡 Step 3: Supervised Synthesis and Recursive Optimization

Once all Helper Agents return their results, the outputs are passed to a Supervising Agent, which:

- Merges, compares, and synthesizes all responses
- Detects redundancies, contradictions, or knowledge gaps
- Identifies emergent patterns and opportunities
- Triggers recursive follow-up actions if deeper inquiry is warranted

This Supervising Agent forms the core of the system’s intelligent feedback loop—enhancing the user’s original request based on learned context, not just answering it.

📺 Step 4: Output Coordination and Display Logic

Final results are routed to the Output Coordinator Agent, which governs how information is presented to the user through flexible, screen-aware display slots. These slots are implemented as dynamic flexbox containers that scale fluidly to fill available screen space across single or multiple monitors. The layout is fully customizable, allowing users to define the size, position, and behavior of each slot based on their workflow needs. To further enhance usability, the primary interaction layer—hosted in the master browser tab—can be looped through and snapped into the same grid. This is achieved by cloning the master tab and integrating it seamlessly into the layout, preserving a clear separation between orchestration controls and content delivery without sacrificing screen real estate. The Output Coordinator handles:

- Slot allocation, ensuring results are displayed in areas that reflect their importance and relevance
- Smooth transitions and display timing, including fade-ins, freezes, and content replacement logic
- Responsiveness and contextual awareness, adapting presentation based on the evolving task flow

- Event-Driven Refresh – New queries, memory updates, or evolving user goals can trigger partial or full re-rendering of display slots
- Topic Transitions – When a user switches topics, previous outputs may fade or be archived, while a “Back” button allows recovery of recent insights
- Time Management – Outputs may persist longer if deemed more relevant, or rotate out smoothly during high-frequency tasks
- The optimization layer and output slots can be manually toggled on or off by the user at any time, allowing for flexible screen usage and focus control. This ensures that advanced coordination features remain accessible without occupying screen space unnecessarily.

🎛️ Core UX Interactions

Each display slot may include built-in controls:

- Freeze – Locks an output to prevent it from being overwritten
- Expand – Reveals more details, context, or reasoning
- Tree-of-Thought View – Displays alternative reasoning paths or similar insights the system considered but filtered

🧑 Human-in-the-Loop Controls

Most importantly, the system is designed to keep the human fully in control. At any time, the user may enter simple natural-language instructions such as:

“Keep this result visible for longer”
“Add a button to export as image”
“Refresh every 10 seconds”
“Hide all diagrams for now”

These textual overrides are processed by the Output Coordinator as part of its reasoning layer and are treated as live interface instructions, allowing seamless interaction without technical configuration.

This ensures that the UI layer remains not only intelligent and adaptive—but transparent, tunable, and collaborative.

🔧 Practical Example: Electrical Installation Business

A professional user—such as an electrician—can define a local folder as a live data source for one or more AI agents. This folder might contain:

- Wiring plans and layout diagrams

- Safety manuals and regulatory checklists
- Contracts, work orders, and customer correspondence
- Historical project notes or troubleshooting records

All files are indexed into a local vector database, allowing agents to retrieve relevant context instantly during reasoning.

When the user initiates a new task—such as drafting a project offer or reviewing a compliance-critical job—the system activates a coordinated AI workflow. Depending on the request, the agents may:

- Extract applicable safety or legal requirements
- Compare the task to previous projects to identify missing steps or recurring issues
- Suggest optimized layouts or resource allocations, based on learned best practices

Tasks can be flexibly distributed across local and external AI agents, with optional masking/demasking, data confiscation, and privacy filters applied where needed—allowing a secure and modular setup tailored to the user's preferences and risk profile.

This hybrid model balances performance, privacy, and adaptability. Simple lookups and compliance checks can run locally, while deeper reasoning or advanced planning can leverage external models—without compromising control or context awareness.

🖥 Local Execution & Privacy-First Design

The entire agent system can optionally be run inside a secure, sandboxed desktop app, giving users full control over:

- API-based cloud LLMs (e.g., OpenAI, Mistral, Claude)
- Self-hosted local LLMs (for partial or full offline operation)
- Private vector storage and memory indexing
- Full GDPR compliance and local data sovereignty

🧠 Summary

The **OpenGiraffe** architecture forms a dynamic AI optimization layer that adapts to task complexity in real time. It empowers professional users—entrepreneurs, engineers, educators—to work with increased foresight, efficiency, and strategic depth.

Instead of waiting for users to ask the “right” questions, the system actively helps surface smarter paths—offering not just answers, but direction.

🧩 *Additional Transparency and User Control via Local Configurable Interface*

To ensure real-time oversight of the AI orchestration process, the system includes an always-active control panel that can be rendered in three flexible ways: as a collapsible panel within the browser extension, as a standalone full-page interface, or as a dynamic display slot. Acting as the system’s 'brain', it continuously visualizes the live reasoning and decision paths of key agents such as the Input Coordinator, Output Coordinator, and Supervising Agent. All three display variants support customizable templates, allowing users to adapt the interface to their specific workflows and preferences. The panel runs entirely on-premise as part of the self-hosted orchestration environment and is fully configurable for advanced use cases.

🧠 *Live Reasoning Inspection and Adjustment*

The interface displays and allows real-time interaction with key system logic, including:

- *Detected goals and subgoals, derived from user intent*
  – Can be manually edited, toggled on/off, or locked to preserve critical objectives during reasoning

- *Generated follow-up questions and reasoning paths*
  – Each can be reviewed, edited, regenerated, or deleted individually

- *Adjustable control fields, such as:*

- *Maximum number of follow-up questions*
- *Trigger conditions (e.g., on input change, rule match, or time interval)*
- *Freeform context prompts (e.g., “Do you have more background info to support this goal?”)*

- *Enable/disable checkboxes for logic modules, output types, or agent roles*
- *Display port toggle for routing specific outputs to predefined display slots (e.g., main area, sidebar, external screen)*

⚙ *Output Control*

The behavior of the Output Coordinator Agent—including display slot logic, refresh behavior, and user interaction settings—can be adjusted here as well. Users may also submit natural-language overrides (e.g., “Freeze this result,” or “Hide diagrams”), which are interpreted live by the system.

⚡ *Optimization Utilities and State Awareness*

- *Quick Optimize buttons can be used to trigger strategic prompt improvements or apply higher-order transformations across the agent graph*
- *The system automatically detects new goals or context shifts, e.g., when starting a new task or input thread*
- *Transitions are surfaced transparently, with the option to manually reset or revert the current reasoning context at any time*

Most of the logic displayed in the control interface is live-bound to the underlying DOM, meaning the change is reflected in real time within the running agent system. Rather than acting as a passive viewer, the user becomes an active manager—guiding the optimizer’s evolving thought process with precision and flexibility.

This control layer is essential for high-responsibility use cases where transparency, explainability, and fine-tuned AI alignment are non-negotiable.

🔍 Context Extension Prompts (with AI-guided gap detection)

To better understand the user's intent, the system includes AI-driven context extension prompts. These are not static fields — they are generated or adapted based on what the system identifies as missing or ambiguous information.

For example:

- If the detected user intent is a request to optimize a workflow but doesn't specify the tools used, the system might ask:
  "Which platforms or apps are involved in this workflow (e.g., Notion, Outlook, Trello)?"
- Or if a goal is stated without timeframe or constraints, it might suggest:
  "Do you have any deadlines, technical constraints, or preferred automation tools I should consider?"

These prompts are designed to:

- Actively identify potential context gaps
- Suggest relevant fields or clarification questions tailored to the task

- *Help the AI refine its understanding of both short-term goals and long-term strategic intent*

The system continuously makes its reasoning process transparent, empowering the user with full control. If the output is not as expected, the user can inspect and adjust the underlying logic in real time. All automatically detected context gaps, follow-up questions, and reasoning steps can be toggled on or off, edited, or overridden at any time—ensuring the system remains aligned with the user's intent.

🔧 *Customizable, LLM-Driven Web Interface*

*The web-based interface is not static—it is **LLM-driven and fully customizable**. That means:*

- The system is designed to support dynamic adaptation of forms, logic triggers, and UI components through configurable profiles tailored to specific domains or workflows (e.g., electricians, lawyers, engineers). Users will be able to create and switch between these profiles to activate relevant reasoning strategies and interface behaviors. In addition to personal configurations, the concept includes a curated library of **vetted community templates** created by domain experts. For more open-ended or cross-domain scenarios, the system will also offer **broad, flexible optimization templates** that apply general reasoning strategies without being tied to a specific field.

- *The underlying logic (e.g., how subgoals are derived or how meta-questions are generated) is **controlled by LLMs**, which enables **semantic flexibility** far beyond static rule-based systems.*

- *Developers or domain experts can* ***extend the HTML interface*** *by integrating more complex reasoning steps, business rules, or specialized input fields.*
- *These* ***custom interface configurations*** *(form templates, goal models, control panels) can be* ***shared with the community*** *or bundled as open components.*

This opens the door to a ***plugin-like ecosystem*** *where:*

- *Anyone can create and publish* ***domain-specific optimizers****.*
- *Communities can evolve best-practice reasoning models* ***collaboratively****.*
- *On-premise deployments benefit from a growing* ***library of modular UI + logic bundles****—without compromising privacy or vendor lock-in.*

🖥 Spatial Distribution & Extended Input Channels

While fully functional on standard laptops, the orchestration system is designed for **workstations with multiple screens** and **VR devices equipped with built-in browsers**, where the extended display space enables clear spatial separation of agent outputs and uninterrupted parallel interaction.

Outputs can also be routed to **external endpoints** like AR glasses or mobile dashboards, allowing findings from one user’s session (e.g., a backend analyst) to be delivered in real time to another (e.g., a field technician).

To support more complex or resource-constrained environments, the system allows multiple **master input tabs** to operate in parallel—enabling input from different sources, users, or subsystems without interrupting coordination.

In addition to standard input methods, the **OpenGiraffe** architecture supports **sensor-level and system-state input**, including:

- Application state snapshots
- DOM structures from web interfaces
- Clipboard, pointer, and voice input
- Sensor feeds from AR/VR devices or robotics platforms

This extensibility allows the system to bridge physical and digital contexts in real time—making it suitable not just for analysis, but also for **situational coordination and cross-device collaboration**.

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

In this use case, a user interacts with the system via a chatbot interface and asks a seemingly simple question: “Can you find me a good USB stick?”

Rather than returning a single product recommendation, the system activates a parallel multi–AI agent workflow that expands the request across multiple dimensions—revealing smarter options, identifying hidden risks, and suggesting long-term strategies the user didn’t explicitly ask for.

The process starts with the Input Coordinator Agent, which interprets the prompt, consults memory, and determines that deeper inquiry is warranted. It generates a set of optimized follow-up questions that break down the request into technical, contextual, and strategic subcomponents.

Each question is then routed to a specialized AI helper agent:

- One agent builds a manufacturer comparison table, evaluating flash type (SLC, TLC), controller quality, encryption, shock resistance, and warranty
- Another matches USB stick types to different user profiles (amateur, professional, enterprise), factoring in durability, write endurance, and portability
- A third explores alternative storage media such as M-DISCs, WORM optical formats, or enterprise-grade immutable storage—surfacing scenarios where long-term data retention is critical
- A visual agent renders diagrams comparing lifespan, usage patterns, and reliability under various environmental conditions

These results are reviewed by the Supervising Agent, which identifies redundant outputs, fills contextual gaps, and ranks relevance. The refined insights are passed to the Output Coordinator Agent, which dynamically arranges the content across screen-aligned display slots—prioritizing clarity and continuity.

As the user continues the session, the system adapts:

- New outputs replace outdated ones or fill available visual slots
- Relevant results remain longer on screen to support decision continuity
- Layout and transitions are managed smoothly to avoid visual clutter

Behind this, the agents rely on global memory—which may include user-provided data such as company size, data sensitivity, access requirements, or past decisions. From this, the coordinator can infer hidden intent structures. For example:

“The user’s company handles internal legal documents and shares access across teams.”

From this, the system might infer and act on secondary objectives such as:

- “Prioritize encrypted or access-controlled storage devices.”
- “Recommend storage with long-term reliability under frequent use.”
- “Suggest a hybrid approach that offsets the weaknesses of single-device solutions.”

In this case, the system might suggest:

“Pair a USB stick for day-to-day use with an archival-grade M-DISC stored in a fireproof case inside a bank locker—ensuring redundancy, physical security, and long-term resilience.”

These types of proactive, meta-level suggestions help the user avoid critical oversights—such as trusting vital data to a device that may fail after 10 years, when alternatives exist that last 100+ years.

🎯 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**. 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.

🔄 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.

🔗 Integration with External Services via Connector Protocols (e.g., MCP)

The orchestration tool can connect to external services—like email, calendar, task management, or CRM platforms—via standardized **connector protocols** . One prominent example is **Anthropic’s Modular Context Protocol (MCP)** , but similar connectors exist across ecosystems (e.g., OpenAI Tools, custom APIs, function calling, or webhook bridges). These connectors act as **adapters**, enabling AI agents to interact reliably and securely with external systems.

🔍 Note: MCP itself is not a memory framework or agent host—it's a **context-passing protocol** that allows structured interaction with LLMs. The orchestration tool is **connector-agnostic** and can work with whichever integration protocol your chosen AI model or platform supports.

🎙️ Multi-Input Architecture with Specialized Roles

The orchestration logic allows **multiple master tabs** to operate in parallel; each linked to a different input modality or task role:

- **Master Tab 1** might connect to a **voice input stream**, used for natural-language commands like:
  - “Display the client follow-up draft in slot 5.”
  - “Show today’s calendar in slot 2.”
  - “Copy the notes from slot 4 into slot 1.”
- **Master Tab 2** may run a **text-based chatbot** for research, Q&A, or strategic reasoning—entirely separate from UI control commands.

This separation enables **fast, non-blocking interaction**, where operational UI actions and deeper reasoning tasks can proceed in parallel.

⚙️ Local LLMs for Low-Reasoning UI Tasks

To minimize API usage and boost performance, **simple orchestration commands**—such as:

- “Show the content from slot 3 in slot 1”
- “Freeze slot 5”

- “Move calendar view to slot 2”

can be processed entirely **by a small, locally hosted LLM** . These lightweight models handle basic intent parsing and slot routing with minimal latency—keeping cloud LLMs free for heavier reasoning tasks. This also enhances privacy and offline usability.

MCP Integration and Context-Oriented Orchestration

The orchestration layer in this system offers powerful support for modular, high-speed, and context-aware automation workflows. A key enabler of this architecture is the integration of external systems through connector protocols like MCP (Modular Context Protocol)—used by Anthropic—and similar interfaces available for other LLM platforms, including ChatGPT's connectors or even custom REST APIs. These connectors allow agents within the orchestration framework to interact directly with external services such as email platforms (Gmail, Outlook), calendar tools (Google Calendar, Microsoft 365), project management apps, or CRM systems. When integrated correctly, the orchestration tool becomes not just a reasoning assistant, but a real-time operational interface.

🧠 Low-Level Reasoning with Lightweight Local Models

Simple routing tasks like "show draft in display slot 1" or "mirror output from slot 3 to slot 2" can be delegated to lightweight, locally hosted LLMs. This reduces latency and ensures that basic orchestration logic remains offline, transparent, and fully under user control.

🔁 Secure Human-in-the-Loop Execution

The orchestration tool does not force autonomy by default. Whether tasks are executed automatically or shown as drafts depends entirely on the template and system configuration. Secure templates prioritize:

- Displaying all automation steps (e.g., email drafts, calendar entries) in a visual display slot first
- Requiring explicit user approval via voice or button before final submission

That said, advanced users can configure fully autonomous flows if permitted by the connected service (e.g., n8n, custom APIs, or specific connector permissions).

🧩 n8n as a Seamless Integration Layer

An ideal companion to the OpenGiraffe orchestration system is n8n — a source-available, browser-based, and locally hostable automation engine. While not required for basic operation, it extends the system’s capabilities significantly by enabling deeper backend logic, integration with third-party tools, and automation of multi-step workflows.

n8n integrates cleanly into this modular OpenGiraffe architecture by:

- Acting as the backend task executor triggered by orchestration logic

- Running predefined automation chains (e.g., send an email, query a CRM, fetch and process a document)
- Responding to display slot routing or contextual instructions from helper agents

Since n8n operates within the browser, it can be opened in its own tab and authenticated locally. Once logged in, orchestration agents (e.g., controlled via user speech, text, or intent) can trigger logic either through DOM interaction or direct API/webhook calls. This creates a bridge between user-level prompts and complex, autonomous backend automation — all under full user control.

🧠 Real-World Example: Alias-Based Orchestration and Real-Time Voice Overrides

The orchestration system intents to allow users to trigger complex, multi-agent workflows using simple aliases — such as:
“Activate workspace747”

These aliases are predefined in a backend automation layer (e.g. using n8n etc.) and can launch full stacks of coordinated actions across agents, reasoning units, UI displays, and logic controllers.

Each element in the system — whether an agent, slot, coordinator, or setting — is assigned a unique identifier, generated from customizable HTML templates that define the structure of the orchestration stack

. These templates include:

- Global and immediate user intent
- Reasoning traces
- Follow-up questions
- UI display slot routing with settings
- Ai agent behaviors

The result is a fully modular interface where every component — from individual control elements and AI agents to entire automation workflows — can be addressed, activated, and routed dynamically. Users can configure the system so that a specific unique ID, alias, or spoken name acts as a trigger. Depending on the setup, this can activate a listening AI agent, reorganize output slots, or execute backend logic — prompting helper agents, coordination agents, or supervising agents to respond accordingly. Even full automation workspaces can be launched in real time via voice, text input, or programmable keys (e.g., Streamdeck). These programmable buttons can also be embedded directly into the user interface through customizable templates."

- "agent 23– rephrase to sound more confident"
- "output_coordinator – expand slot_5 and highlight key terms"
- "reasoning_intent_1 – I search for the firmware not for the bug"
- "supervisor_22 – trigger a steeprocketsearch747(alias) why I can't burn the MDISC"

Voice commands can act as a parallel master control layer, enabling overrides, adjustments, or clarifications even while automated flows are running. Since everything is driven by backend automation and HTML-defined identifiers, the entire stack remains fully configurable, extensible, and reusable — making the system highly adaptable for different users, workflows, and use cases. The **OpenGiraffe** architecture supports parallel input streams with unified or separate backend automation, where display slots—independent of the active master tab—can be dynamically managed by AI using simple logic patterns or manually assigned by users to specific tasks for flexible, context-aware layouts.

🎤 Multi-Device Voice Input Integration

Voice commands are intended to be captured through a lightweight voice interface embedded in multiple endpoints:

- A **mobile app** (smartphones/tablets) with push-to-talk or passive listening
- **AR headsets** or wearable devices equipped with microphone access
- A built-in module within the **orchestrator app itself**, allowing direct control via **headset or microphone** input while working in the browser

🧬 Intent-Triggered Automation: Passive AI Detection

In more advanced usage, no direct prompt is even necessary. The orchestration system can passively infer user intent from contextual activity. For example:

A user records a meeting through a master tab interface.

- The system detects this is a structured team session and triggers:
    - Real-time transcription (via Whisper)
    - Context extraction: goals, risks, unresolved questions
    - Summarization and optimization suggestions
    - Gap and opportunity detection
- These results are compiled into a structured PDF overview.
- The system drafts an email containing the meeting summary, PDF, and personalized follow-ups to each participant.
- The draft email is automatically displayed in an available display slot for review and confirmation.

This entire chain can be executed without cloud dependency — running locally on user-owned infrastructure or in a private n8n instance, preserving full data control and privacy.

🚀 Summary: Dynamic Human-AI Collaboration

The **OpenGiraffe** architecture redefines AI integration by enabling real-time, context-driven orchestration across modular agents — all triggered through speech, text, or detected behavior. By leveraging n8n's automation layer in combination with **Optimando's** orchestration logic, users gain an interactive, agent-driven environment where even highly complex workflows become intuitive and auditable.

Instead of isolated chatbots or opaque automations, the user sees exactly what happens — who does what, when, and why — and remains in control at every step.

📷 OpenGiraffe & QR: QRGiraffe →From Static Codes to Orchestrated Intelligence

QR Codes are everywhere—but their default behavior is outdated and risky. Most systems execute QR payloads instantly: opening links, launching apps, or triggering silent actions. No inspection, no consent, no explanation.

QRGiraffe redefines this paradigm.

Instead of treating QR Codes as blind triggers, it transforms them into structured, intelligent entry points—processed securely, explained visually, and orchestrated through modular automation logic.

🧠 What Happens When a QR Code Is Scanned in **QRGiraffe**?

Local Capture, No Execution

- Upon scanning, the QR Code is parsed, encrypted, and stored locally within the **OpenGiraffe**-enabled app.
- → No automatic execution of the payload takes place.
- → No outbound network request is triggered by default.

Explicit User Whitelisting

For a QR payload to proceed, it must be explicitly approved by the user.
There is no automatic trust decision.
Optionally, the OpenGiraffe community can highlight that a source is known or cryptographically signed, but approval is always manual by the user itself. (Explicit Whitelisting)

Encrypted Orchestration Trigger

Once the orchestrator is online, the payload is sent securely for analysis.
The orchestrator activates a predefined workflow and the general optimization layer—completely local and privacy-respecting.

Visual Analysis & Contextual Insight

Unlike traditional QR tools, **OpenGiraffe** uses the entire screen real estate to deliver a rich user experience:

- Full-page preview of destination intent
- Visual overlays explaining actions, risks, comparisons, suggestions, gap detection
- Structured metadata view
- Step-by-step automation reasoning
- Separate panels for content, trust signals, and suggested actions

The QR result is no longer a black box—it’s an interactive, explainable environment, fully under user control.

Form Handling Policy: Preview Only

OpenGiraffe never auto-submits forms. If a QR payload includes a form or the optimization layer pre-fills a form for convenience, it is parsed and displayed in preview mode. Actions like submission, authentication are only possible after explicit user confirmation.

📨 Enhanced QR Experiences for Trusted Providers

WR Code providers can include structured instructions to display a visually organized overviews within the orchestration interface and provide additional data. (Workflow Ready Codes)

- ✍ Digital signatures to prove authenticity
- 📄 Structured metadata (validity, purpose, automation flags)
- 🧠 Semantic intent layers (Simple text instructions for AI-Agent automation), such as:
  - “Show additional video”
  - “Draft an Email with this content and book a meeting in calendar as draft”
  - Open a signing interface (e.g., DocuSign) → Perfect for events, delivery handoffs, or mobile B2B transactions
  - “Display a visually organized discount overview”
  - “Show a 3D model in a separate slot”

🎯 Custom automation hooks for safe orchestration triggers

These enhancements help OpenGiraffe present a smarter, richer user experience—without compromising trust or security.

🔐 Security Architecture: Groundbreaking by Design

QRGiraffe introduces multiple protective layers by default:

- ❌ No execution without consent
- ✅ User-controlled whitelisting
- 🔍 Transparent orchestration flows
- 🧩 Modular reasoning for every action

For advanced users, optional isolation layers are available:

The orchestrator can be run inside a dedicated virtual machine, booted from an external SSD, using a hardened operating system. This provides full execution separation from the host, useful for **high-risk environments or forensic workflows** . RAM-only operating systems offer extreme execution isolation by eliminating all disk-level traces, even within virtualized environments.

🖥️ Designed for Full-Screen Orchestration

Unlike mobile QR scanners or browser redirects, the **QRGiraffe** concept uses every pixel to explain, visualize, and coordinate WR-based workflows.

- 🧠 Reasoning panels alongside content previews

- 🛠 Interactive controls for reviewing each step
- 🔍 DOM overlays, metadata inspection, automation logs
- 📊 Risk and benefit indicators and origin validation
- 🧩 Adaptive layout for multi-monitor or widescreen displays

The result: a truly intelligent WR interface—where the scan is just the beginning of a controlled, explainable process. (WR Codes are Workflow Ready Codes based on the QR40 Code technology)

Optional Integration with Distributed Ledger Technologies (e.g., IOTA)

- **QRGiraffe** includes a conceptual mechanism for secure, decentralized logging and verification using distributed ledger networks such as IOTA.

→ Payloads such as public URLs, structured metadata, or AI automation instructions can be selectively published or verified via the Tangle.

→ Only publicly shareable, non-personal information is eligible—no personal data, user identifiers, or location-based metadata are stored.

→ Anonymity is preserved by design, with no technical linkage to individuals, organizations, or geographic locations unless explicitly configured.

→ Users remain in control and may activate this feature as needed by supplying their own API keys, node URIs, or access credentials.

🔧 Example Use Case

A WR Code triggers a process. The process is hashed client-side (SHA256(input+timestamp)), stored locally, and optionally logged on IOTA for verifiability. No content or identity is revealed. → ✅ Compliant & privacy-safe.

Example 1 – Claiming a Limited Offer

A user scans a WR Code on a product flyer to claim a limited-time discount.
Instead of directly opening a URL, the system:

- Parses the QR data (e.g., offerID=SUMMER10)
- Hashes it with a timestamp: SHA256(SUMMER10 + 2025-07-02T12:03:00Z)
- Stores the hash locally (e.g., in browser or app storage)
- Optionally logs it on IOTA to prove the claim attempt—without exposing user data

✅ **The user stays anonymous, and the offer can be verified without tracking or profiling.**

Example 2 – Accessing a Secure Building or Event

A visitor receives a QR-based invitation to a co-working space or conference.

- **Upon scanning, the data is hashed client-side with a timestamp**
- **The hash is compared against a locally cached or distributed whitelist**

- If valid, the access system grants entry

✅ No personal information or credentials are transmitted—only a cryptographically verifiable claim is processed.

Example 3 – Privacy-Preserving Loyalty Points

A customer receives a QR receipt with embedded loyalty points.

- The QR is scanned, hashed locally, and stored
- On future visits, the customer re-presents the hash for point redemption
- Optional anchoring to IOTA ensures tamper-proof proof of claim

✅ Users never register or log in—loyalty remains anonymous, yet verifiable.

Example 4 – Secure Device Pairing

A QR on a new IoT device (e.g., router, sensor) is scanned to start onboarding.

- OpenGiraffe processes the pairing request locally
- The payload is verified without external communication
- Optionally, a hash of the process is stored for future auditing

✅ Devices are added securely without sending sensitive setup data to third parties.

Forward-Looking Vision: Enabling Zero-Knowledge Verification

Although not yet implemented, the architecture is designed to support zero-knowledge proofs (ZKPs) in future versions.

In this model, the system could:
• Coordinate local proof generation (e.g., "I am over 18", "I hold a valid credential")
• Accept and verify ZK proofs without revealing any underlying data
• Trigger logic based on verifiable claims — without ever seeing who the user is

✅ **Even in these advanced use cases, the user's identity is never exposed to OpenGiraffe or any of its services.**
✅ **All sensitive operations occur locally and client-side, maintaining strict privacy by design.**

Extendable to Other Domains Beyond QR

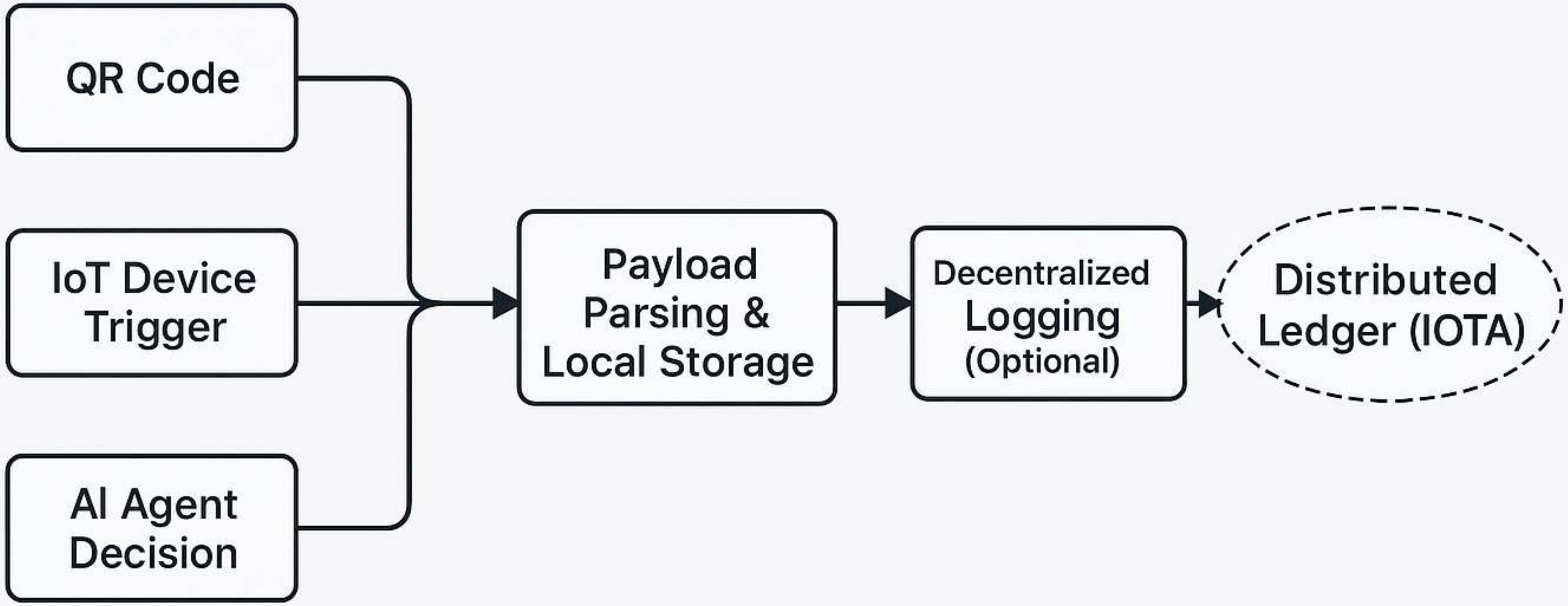
This decentralized verification approach is not limited to QR interactions.

→ It can also be applied to other components of the **OpenGiraffe** system, such as:

- IoT device triggers or smart home automation routines
- AI agent decisions and reasoning chains
- Audit trails for orchestrated workflows or inter-agent communication

→ **This enables cross-domain consistency, verifiability, and long-term integrity of automation processes in dynamic and distributed environments.**

OPTIONAL DECENTRALIZED LOGGING BEYOND QR INTERACTIONS


QR Code
IoT Device Trigger
AI Agent Decision
Payload Parsing & Local Storage
Decentralized Logging (Optional)
Distributed Ledger (IOTA)

🔗WR Code Security and Local-First Orchestration in OpenGiraffe

To enhance security, integrity, and privacy in AI-driven automations, the **OpenGiraffe** framework introduces a local-first execution model where every QR Code is treated as potentially unsafe by default. Each scanned **WR Code** is displayed to the user in readable form before any action is taken, ensuring that no AI-triggered automation is executed without explicit user confirmation.

Users can make informed decisions when scanning QR Codes from well-known providers, even if those codes are not explicitly whitelisted by the community. Public whitelists and individual trust decisions can coexist, giving users full flexibility and control over execution.

To further strengthen security, **OpenGiraffe's** orchestration templates include automated checks for phishing risks and other potential threats before executing any **WR Code** actions. These security templates are fully configurable using simple, human-readable language, enabling both technical and non-technical users to define custom safety rules for AI automations.

This approach is particularly valuable in the context of AI-triggered automations. Unlike traditional WR Code uses—where malicious codes typically result in simple redirection or phishing—AI-driven automations carry higher risks. A spoofed **WR Code** could trigger unauthorized actions or malicious workflows without user awareness. It is important to highlight that WR Codes are already exploited in the real world for malicious purposes, including phishing attacks and malware infections. Cases have been reported where altered **WR Codes** on shared vehicles or public terminals were used to deploy malware or steal sensitive information.

✅ Summary: Trust, Clarity, Control

With QRGiraffe, QR Codes are no longer a risk—they're an opportunity to activate intelligent, modular workflows with:

- 🧠 Full semantic understanding
- 🛡️ Deeply layered protection
- 🖥️ Visual orchestration across the multi-screen setups
- 🔗 Optional trusted provider integrations
- 💡 Future-ready ZKPs, adaptive architecture

QRGiraffe transforms static QR Codes into dynamic, orchestrated user experiences—intelligent by design, controlled by you.

📍 Geofence-Orchestrated Intelligence

Privacy-Preserving Location Triggers with Zero-Knowledge Verification and Intent-Aware Mobile/AR Automation

OpenGiraffe introduces a new paradigm in geofence-driven automation: combining **zero-knowledge proofs (ZKPs)**, **on-device geolocation triggers**, and **intent-aware AI workflows**—without reliance on cloud infrastructure or surveillance-based tracking. Unlike traditional geofencing tools, which often link location to identity and upload data to centralized servers, **OpenGiraffe** handles detection, verification, and orchestration either **locally or within a trusted private backend**, making it ideal for **wearables, AR/AI Glasses, and smartphones**.

🔒 Zero-Knowledge Entry Proofs – No Identity, No Coordinates, No Surveillance

As a user approaches a defined physical location (e.g., a logistics gate, event zone, or facility entrance), their device detects entry via GPS, Bluetooth, or Wi-Fi. Instead of revealing precise coordinates or user identity, **OpenGiraffe** locally generates a **cryptographic hash**:

SHA256(zoneID + ISOtimestamp)

This functions as a **proof of presence**—optionally anchored on a ledger like IOTA—but does **not disclose who the user is**, or where exactly they are. This makes geofencing usable for sensitive environments (e.g., factories, labs, field operations) while remaining fully privacy-preserving.

Example: Privacy-Preserving Retail Offer with Anonymous ZKP Verification

A retail store offers an exclusive online discount that is only accessible to users who have physically visited the store (and maybe even scanned a WR Code additionally). The process is built to ensure complete privacy—without any GPS tracking or identity linkage.

How It Works:

1. When the user physically enters the store, their smartphone detects presence locally—via geofencing, GPS, or WR Code. (visit a local event or representation first)
2. The phone generates a Zero-Knowledge Proof (ZKP)—a cryptographic hash confirming that the event occurred.
3. This **hash is:
    - Stored locally on the device (e.g., in the user's storage).
    - Optionally anchored on a public blockchain (e.g., IOTA) for verifiable, tamper-proof proof of presence.

✅ No personal data is collected.
✅ No GPS tracks, no timestamps, no identity—only an anonymous presence hash exists.
✅ The store has no access to any data at this stage.

4. Later—when the user chooses to buy online—the retailer's system can:
   - Ask the user to provide the presence proof (the stored hash).
   - Compare it against the public IOTA anchoring to verify eligibility.
5. If the hash matches, the user is whitelisted for the exclusive offer.

- The retailer only sees the valid proof—but has:
  - ❌ No knowledge of who visited.
  - ❌ No knowledge of when they visited.
  - ❌ No way to track or correlate users to physical behavior.

If the user decides to purchase, they may naturally provide PII during checkout (name, email, shipping)—but this remains decoupled from the original physical visit.

✅ Key Privacy Advantages:

- No tracking: Geofence detection stays entirely local—nothing is transmitted when presence is detected.
- Anonymized proof only: The vendor sees only a hash—never location, time, or identity.
- User-controlled verification: Only the user can later choose to reveal a valid proof to access the online benefit.

In short:
👉 Physical presence is provable, but not traceable.
👉 Vendors can verify "someone" was there—but never know who, when, or where exactly.
👉 Users control both timing and identity disclosure separately.

📦 OpenGiraffe & WR Code Automation Sharing: Unlocking the Power of QR40 for Portable, Privacy-First Workflows

While **QRGiraffe** has already introduced a new era of secure and explainable QR interactions, an equally powerful capability is the ability for **OpenGiraffe** to generate QR40 **"Workflow-Ready**

Codes(WRCodes)” for seamless automation sharing. This approach enables entirely new forms of portable, decentralized, and context-aware automation delivery—without reliance on centralized services or invasive tracking.

Importantly, WR Codes generated by **QRGiraffe** utilize standard QR Code technology (ISO/IEC 18004). This is not new technology in itself—the innovation lies in the unique way this existing technique is utilized to enable secure, explainable, and privacy-respecting automations. The term "**WRCode**" (Workflow-Ready Code) is used descriptively to highlight the workflow-readiness of the generated WR Codes. It does not represent a new code format or suggest any modification of the underlying WR Code standard—it is simply a conceptual layer on top of globally recognized WR Code technology.

In addition, WR Codes can be **dynamically generated** by digital services, software agents, or even robots—allowing real-time, system-agnostic deployment of automation workflows across diverse physical or virtual environments

🔄 From Static Codes to Dynamic, Shareable Automations

With QR40 WRCodes, users can:

- Export complete automations in WR Code format (within size limits, minified and optionally compressed)

- Include key initial inputs or settings within the QR
- Trigger automation that is fully personalized through local PII (personally identifiable information) already stored in the user's OpenGiraffe instance
- Enable the automation to explicitly request missing input from the user before execution

The result is a truly intelligent QR system that not only shares information, but shares action—empowering users to execute meaningful processes immediately, while retaining full control.

🛠 How It Works:

1. Automation Export:
   - Any workflow designed in OpenGiraffe can be serialized, minified, and embedded into a QR40 WRCode (up to ~3KB practical size).
2. PII-Based Personalization:
   - Upon scan, the orchestrator uses locally stored PII (name, department, preferences, saved accounts) to pre-fill the automation with user-specific data—without any server dependency.

3. Contextual User Input:
    - If required inputs are missing, the orchestrator can prompt the user dynamically (e.g., quantity, reason, comment)—ensuring the automation has all it needs before execution.
4. Controlled Execution:
    - Execution remains fully transparent, previewed, and user-approved in line with QRGiraffe's strict security philosophy.
5. Optional Anonymous Logging with IOTA & ZKP:
    - Events triggered by WRCodes can be optionally hashed and logged on the IOTA Tangle for decentralized verification.
    - Zero-Knowledge Proofs (ZKP) can be employed to prove that an event occurred or a condition was met without revealing any sensitive information.
    - Geofencing data can be included in these verifiable events fully anonymously, enabling location-based automation triggers without exposing exact positions or user identity.

🎯 High-Value Use Cases for WR Code (QR40) Workflow Sharing:

Use Case	Description	Value
📦 Field Inventory & Maintenance	Technician scans WR on equipment → automation pre-fills with stored user data + prompts for piece count/status → generates instant report	Saves time, reduces errors, offline-capable
📝 Legal Document Automation	WR on contract triggers automation that pulls in stored company/user PII → asks for missing fields (e.g., delivery date) → outputs ready-to-sign doc	Accelerates legal/admin processes while ensuring compliance
🎁 Coupon & Offer Claims	Marketing flyer with WR triggers personalized coupon claim → automation uses stored user data + asks for optional preferences → generates offer	Drives engagement without forcing app installs or logins

🚧 Access Control & Visitor Management	WR-based access request triggers automation that uses stored PII ➔ asks visitor for reason/ETA ➔ processes entry approval offline or hybrid	Boosts security, reduces friction, maintains privacy
🚗 Automotive & IoT Pairing	Scanning WR on device starts onboarding automation ➔ local data pre- fills known fields ➔ user adds any missing details ➔ device is securely paired	Simplifies device setup without cloud reliance
📍 Geofenced Event Automation with Anonymous Logging	WR at physical location triggers time - or location-sensitive automation ➔ event is logged anonymously on IOTA ➔ optional Zero-Knowledge Proof ensures claim validity	Enables privacy-preserving location-based workflows and verifiable event logs

💡 Strategic Advantages:

- Privacy-First by Design: No personal data is embedded in the QR—only the automation logic or references.
- Offline & Resilient: Automation execution is possible even without an active internet connection.

- Portable & Shareable: QR40 WRCodes can be printed, shared digitally, or embedded in products, without any back-end dependency.
- Time-Saving & Error-Reducing: Pre-filled forms, context-aware prompts, and automation minimize manual errors and speed up processes.
- Verifiable Without Tracking: Optional IOTA and Zero-Knowledge Proof integration allows for cryptographically verifiable events without compromising user anonymity.

🚀 Real-World Impact:

By enabling the creation and execution of portable, personalized automations through QR40, OpenGiraffe empowers:

- Business workflows that require zero infrastructure.
- Customer interactions that respect privacy and reduce barriers.
- Industry use cases where security, speed, and flexibility are paramount.

The QR becomes more than a link—it becomes an actionable gateway to orchestrated intelligence.

For organizations, event organizers, field services, and many others, this capability creates a new automation frontier: one where privacy, control, verifiability, and intelligence are not trade-offs, but standard features.

Security Concept for PII Protection and Automations in OpenGiraffe (QR-Giraffe Module)

1. Objective

OpenGiraffe follows a strict privacy-by-design approach to protect users from identity theft, IP theft, extortion, and data misuse. This security concept ensures that personal identifiable information (PII) and other confidential data remain fully under the user's control at all times.

2. Local Encrypted Vault

- PII and secrets are stored exclusively in a locally encrypted vault.
- Encryption used: AES-256-GCM or comparable secure methods.
- By default, highly sensitive PII (such as names, addresses, financial data) is never prefilled, displayed, or exposed during automated processes. The insertion of PII into any form occurs only at the exact moment of manual user confirmation, - To support cases where AI-generated entries or complex automations are involved, an additional option allows users to temporarily

decrypt and preview specific non-PII content through a secure visual overlay layer prior to submission. This allows verification of dynamically generated data while maintaining strict separation from sensitive PII.

- Decryption and transmission of any data take place only on-the-fly and under explicit manual consent.
- To further strengthen control, the Vault supports adaptive authentication mechanisms based on security level:
    - For routine low-risk automations (Trust Levels 1 & 2): temporary automation access may be allowed without friction, provided no PII is involved.
    - For PII insertion, Vault modifications, identity resets, or any sensitive operation (Trust Level 3): biometric authentication (e.g., Face ID, fingerprint), passphrase, or hardware token is mandatory.
- Biometric authentication is particularly suited for frictionless yet secure experiences on smartphones, which represent the majority of WR Code use cases.
- Users without biometric capability may use a secure fallback such as a PIN code or hardware security key.

- The PII Vault is designed to evolve into a broader personal credential manager, capable of securely storing passwords and authentication tokens. This future capability will allow users to automate sign-up processes, logins, and password handling within orchestrated workflows. While not all features will be available from day one, the Vault is conceptualized as a central pillar for secure identity management and seamless automation. This capability is part of the long-term vision and may not be available in early versions of the system.
- To further enhance security and future-proof the Vault against emerging threats such as quantum computing, the Vault can optionally be hardware-bound through a device-specific fingerprint combined with a cryptographic IOTA-based hash. This mechanism ensures that even if the encrypted Vault is exfiltrated, it remains unusable on any other device due to a mismatch between the stored hash and the hardware identifier.
- Backup and recovery operations involving the Vault are restricted to the mounted and authenticated state only, ensuring that backup passwords or secrets cannot be extracted in offline scenarios. This design makes "Harvest Now, Decrypt Later" attacks infeasible, as the Vault remains cryptographically and physically tied to its original secure execution environment.

3. Network Access and Exfiltration Protection

- Outgoing network traffic remains strictly controlled and policy-driven. For sensitive automation phases, temporary network blocking or user-confirmed release can be enforced to ensure that no unauthorized or automatic data transmissions occur. This flexible approach allows the system to maintain a high level of security while avoiding unnecessary disruption to legitimate use cases where connectivity is essential.
- Users retain full control over provider trust relationships through a local whitelist, which can be modified or revoked at any time.

4. Authenticity and Integrity of WR Codes and Templates

- Every WR Code and automation template is securely verified through an IOTA-based cryptographic hash.
- Templates without a valid signature or with altered structure are strictly not executed.
- Once a WR Code provider is whitelisted, it cannot alter automation templates without risking automatic delisting. Only explicitly trusted providers may be permitted to update templates when necessary.
- This restriction is particularly important for dynamically generated WR Codes, such as those displayed on digital screens or delivered over the internet, where the underlying automation

templates are created dynamically. Such dynamic operations are only allowed for fully trusted and explicitly whitelisted providers.

- Even in these cases, every dynamically generated automation template can be securely logged on IOTA, allowing users to verify and prove at any time that a specific automation was issued by a specific provider at a specific point in time.
- Importantly, any shared automation can be enhanced, adapted, or overwritten directly on the user's device. This flexibility ensures that workflows remain adaptable to specific needs, contexts, and preferences while maintaining security and control. Users are not locked into fixed automations and can modify or fine-tune processes to achieve the best possible outcome.
- Additionally, user-adapted workflows can be stored locally so that repetitive QR scans of the same automation will automatically use the adapted version instead of the original. This enables a continuous optimization of recurring processes while preserving full user control over the automation behavior.
- Every WR Code and automation template is securely verified through an IOTA-based hash.
- Templates without a valid signature or with altered structure are strictly not executed.
- The integrity of every automation is documented through tamper-proof IOTA logs, providing forensic evidence in case of disputes.

- Importantly, any shared automation can be enhanced, adapted, or overwritten directly on the user's device. This flexibility ensures that workflows remain adaptable to specific needs, contexts, and preferences while maintaining security and control. Users are not locked into fixed automations and can modify or fine-tune processes to achieve the best possible outcome.
- Additionally, user-adapted workflows can be stored locally so that repetitive QR scans of the same automation will automatically use the adapted version instead of the original. This enables a continuous optimization of recurring processes while preserving full user control over the automation behavior.

5. Adaptive Trust Levels

To further strengthen trust and security, WR Code implementations may optionally support Zero-Knowledge Proof (ZKP) based qualification checks. This would allow automation templates to require verifiable but privacy-preserving proofs from users or devices before execution. Such proofs could demonstrate:

- Eligibility (e.g., minimum age or verified identity) without disclosing sensitive information,
- Compliance with security policies or certifications,
- Device integrity or vault status.

This optional qualification layer enhances the system's ability to meet regulatory requirements and supports high-trust applications without compromising user privacy or control.

OpenGiraffe uses a multi-level security model to flexibly secure various use cases:

Trust Level	Description	Examples
Level 0: Air-Gap Mode	No network, no PII transmission	Offline access control
Level 1: Manual-Only Mode	Non-sensitive automation only, no PII	Scooter sharing onboarding
Level 2: Trusted Mode	Automation after prior user approval, limited PII	Recurring services
Level 3: High Security Mode	Sensitive processes with mandatory biometric, hardware key or passphrase, ensuring that any release of PII requires explicit user confirmation	Contractual or IP-sensitive tasks

6. Tamper Protection and Transparency

- All automations undergo a policy and template verification before execution.
- Outgoing network activities are blocked or explicitly controlled during sensitive phases, ensuring that PII is never prematurely exposed.
- By separating data visualization for non-sensitive content (via secure overlays) from actual data transmission and PII handling, OpenGiraffe ensures that sensitive information cannot be silently extracted by malicious templates or external scripts.
- Users receive visible control options and can review, track, and revoke automated decisions at any time.

7. Governance Considerations (Informal)

WR Code is designed as an open and community-driven standard that defines how QR automation templates can be created, verified, and transparently shared. This standard operates independently of the OpenGiraffe software, which remains fully open source and is not affected by any validation or governance process.

To support integrity, trust, and transparency, wrcode.org serves as a public validation platform where organizations can voluntarily authenticate themselves and register their QR automation templates.

This process allows templates to be made publicly visible, tamper-proof, and immutably logged—using IOTA—ensuring that:

- Templates can be proven to belong to a specific verified organization.
- Template code remains public, transparent, and resistant to manipulation.
- End users can verify the legitimacy of automation templates before use, and in case of fraud attempts, all interactions can be forensically traced and proven.

This approach allows users to freely choose how they interact with WR Codes, while companies can opt into higher validation standards to build trust and visibility. Participation in the validation process is voluntary, ensuring that the decentralized and open character of the WR Code standard is preserved.

Future governance may evolve to further define the fair use of the WR Code ecosystem as adoption grows.

WR Code is designed as an open and community-driven standard. While the system encourages broad adoption in both closed and open environments, it is intended to remain associated with transparent, privacy-preserving automation principles as implemented in OpenGiraffe.

Future governance may evolve to further define the fair use of the WR Code ecosystem as adoption grows.

8. Conclusion

Looking ahead, WR Codes may also gain relevance for emerging technologies such as AR glasses and wearable devices, where quick, secure, and standardized automation triggers will become increasingly valuable. The open nature of the WR Code standard ensures that it can flexibly adapt to such future use cases while maintaining a strong focus on privacy, interoperability, and user control.

This security concept ensures:

- Maximum user control
- Verifiable integrity of all automations and QR interactions
- Protection against auto-pull, manipulation, and exfiltration

OpenGiraffe not only protects data but also empowers users with digital self-determination against modern threats.

Open Use, Verifiable Trust, Confidentiality, and Secure Interoperability with WR Codes

WR Codes are based on the globally recognized QR Code Model 40 standard, part of the international ISO/IEC 18004 specification. While "QR Code" is a registered trademark of Denso Wave Inc., the underlying technology has been made freely available for both commercial and private use worldwide.

WR Codes do not alter the technical structure of QR Codes but introduce a new way to use standard QR Codes as triggers for verifiable, flexible automations. This enables any individual, developer, or organization to create and use WR Codes entirely within private or internal infrastructures, without mandatory registration or licensing. This ensures fast, cost-effective automation while maintaining full privacy and control over internal processes.

Building Trust through Verifiable Automation

In cases where WR Codes are used in public-facing services—for example, customer onboarding, mobility apps, or hospitality—trust, transparency, and legal defensibility become essential. To meet these needs, WR Codes can be optionally registered at wrcode.org:

- Each WR Code is linked to a tamper-proof cryptographic hash stored immutably on decentralized ledgers such as IOTA.
- This allows end users to verify that the automation triggered by the WR Code matches the registered, untampered template.
- The original automation template must also be locally accessible and unmodified at the time of execution, ensuring that both hash verification and content verification are possible.

This dual verification ensures that users can:

- Confirm the authenticity of the automation,
- Audit and prove the exact process that was executed,
- Build trust in customer-facing services while ensuring compliance with legal and security standards.

The wrcode.org registry is centrally governed to enforce ethical use, prevent abuse, and protect the integrity of the system. Registrations may be declined or revoked if policies are violated.

Confidential Automations: Optional Disclosure, Always Verifiable

In high-risk operations—such as sensitive industries, regulated environments, or security-critical contexts—organizations may choose to keep their automation templates undisclosed while still maintaining full verifiability:

- The cryptographic hash of the automation is still registered,
- The original automation template remains local and private, yet can be verified against the hash.

This ensures that while the content of the automation remains confidential, its execution integrity is provable. End users interacting with such WR Codes are informed when a process is undisclosed and can decide whether to proceed. This preserves both privacy and user choice.

Full Interoperability and User-Controlled Data: The PII Vault (Dual Use)

For advanced use cases involving cross-device continuity, personal data reuse, and AI-powered automation, the Dual Use License introduces full interoperability through the OpenGiraffe Orchestrator and QRGiraffe tool.

At the heart of this system is the PII Vault—but critically:

- The PII Vault belongs to the user, not to the publisher of the WR Code.
- The publisher (e.g., hotel, mobility service) may design WR Codes that request data to trigger specific automations.
- The user holds and controls this data inside their own private PII Vault, which functions as a kind of personal key to public automations.

The PII Vault allows:

- Customizable, structured data storage—identity, preferences, transaction history, or any user-defined information,
- Data to be shared only when the user consents, with full transparency and security,
- **Context-Aware Automations with Dynamic WR Code Generation**

Public WR Codes can dynamically adapt not only the triggered automation but also the visual representation of the WR Code itself—based on context data stored in the user's private vault or derived from non-PII environmental signals. In selected scenarios, context can influence both the execution **and** the generation of the WR Code in a continuous feedback loop.

For example, a user wearing AR glasses might see a WR Code overlay that is generated in real time based on non-personal contextual factors such as personal preferences, time, or device state. This enables highly personalized, adaptive interactions without exposing sensitive information. The dynamically generated WR Code can be selected from a predefined set of registered templates or fully customized to match the specific situation.

All generated WR Codes remain **tamper-proof and verifiable** through cryptographic methods, ensuring transparent and secure automation triggers. Where appropriate, dynamic WR Codes can adhere to emerging **standards or ethical guidelines**, particularly in high-risk or sensitive environments, to maintain public trust and prevent misuse.

This concept is currently intended for **research purposes**, exploring how context-aware visual codes could enable seamless, privacy-preserving automation across devices, AR systems, and connected environments.

This model ensures:

- Privacy-first interoperability,
- Frictionless onboarding without repetitive data entry,
- Security and legal compliance at every interaction point.

Private Use vs. Ecosystem Integration

Organizations that prefer to remain fully private can still use WR Codes within closed, self-contained systems without registering at wrcode.org. After all, WR Codes are only QR Codes that follow a specific standard guideline. These individual setups may involve internally managed automation templates and even private equivalents of the PII Vault.

However, such systems will not be interoperable with the global WR Code ecosystem, nor will they benefit from public trust, cross-device continuity, or ecosystem-driven AI orchestration.

Convenience in the Hotel with WRCode

Easy Check-In

Useful Room Info

Order with WRCode

Quick Check-out

QR Code technology has become a universal tool for bridging physical and digital experiences. Its simplicity, accessibility, and global adoption have made it indispensable across industries. While QR Codes themselves are technically harmless, there have been instances where malicious actors have used QR Codes to direct users to harmful websites or fraudulent services, leading to device infections, phishing attacks, or unauthorized data collection. These incidents underline the need for a secure, transparent framework for any future use of QR Codes in triggering trusted automations.

The **WRCode** represents this next step: a secure, standardized approach to trusted automation triggers, defined and governed by **wrcode.org** .

Defining the WRCode

The concept of embedding AI automation instructions directly within the text of QR Code Model 40 (QR40) was first introduced by **wrcode.org**. This innovation allows automation templates to be carried securely within the QR Code itself, forming the foundation of a new verifiable standard for on-demand digital actions.

A **WRCode** remains fundamentally a standard QR Code based on the ISO/IEC 18004 specification. The underlying QR technology is unchanged.

What differentiates a **WRCode** is strict adherence to:

- Security and operational guidelines defined by **wrcode.org** .

- Tamper-proof automation templates registered and anchored on IOTA.
- A verifiable ecosystem that ensures public trust, issuer accountability, and transparency for customer-facing automations.

In summary: **while every WRCode is technically a QR Code, only those QR Codes that conform to the WRCode guidelines qualify as WRCode.**

The WRCode Security Framework

Access to PII stored within the Vault is strictly denied unless the WRCode automation is verified in real time through an active internet connection. This is essential to validate the tamper-proof zero-knowledge proofs of the software stack, the WRCode issuer, and the device fingerprint anchored on the IOTA blockchain. Without this verification, no sensitive automation involving PII can be executed.

In scenarios where users scan standard QR Codes or WRCode automations while offline, the system will operate in a “non-sensitive” mode: QR Codes can still be scanned and analyzed, and informational content can be displayed, but no AI automation instructions from QR Codes will be triggered, and no PII stored in the Vault will be accessed or released.

This approach ensures that WRCode-compliant applications remain fully functional for general QR Code scanning and offer safe, controlled environments for analyzing standard QR Codes without compromising the strict security and privacy guarantees of WRCode automations. Full WRCode functionality, including AI automation execution and access to sensitive data, is only permitted once the system confirms the authenticity of the entire software and identity chain through **zero-knowledge proof online verification** .

To ensure authenticity and prevent misuse, all Vault applications and **WRCode**-compliant software must include a **mandatory WRCode Validator**. This built-in validator checks whether a scanned QR Code conforms to the **WRCode** standard by verifying its registration, the anchoring of its automation template IOTA, and the validity of the issuer identity through **wrcode.org**.

Only QR Codes that successfully pass this validation are permitted to trigger sensitive automations or request access to personal data stored in the Vault. Non-**WRCode** QR Codes may still be processed by compliant software, such as for analyzing offers or reading non-automated content, but these cases must never access or unlock Vault-stored PII.

This mandatory **WRCode** validation ensures that users and organizations can independently verify whether a code is a genuine **WRCode**, providing a consistent layer of trust and security across all implementations.

The **WRCode** standard is built upon a clearly defined set of security and operational guidelines established by **wrcode.org**, which together ensure the integrity, trustworthiness, and transparency of all **WRCode**-based automations. These guidelines include:

1. **Mandatory Verifiability:** All **WRCode** automations—whether public-facing or internal—must be verifiable through **wrcode.org**. For customer-facing **WRCode** automations, the corresponding automation templates must also be transparently published unless there is a justified reason for confidentiality. In cases where templates remain confidential, their existence must still be provably verifiable through cryptographic anchoring on the **IOTA blockchain**. Additionally, issuers opting for confidentiality must provide a **public statement** through **wrcode.org** explaining the nature of the confidentiality, ensuring that users and stakeholders are aware of the verifiable, yet undisclosed, status of the automation template.

2. **Tamper-Proof Automation Templates:** Every automation template must be cryptographically hashed and anchored on the **IOTA blockchain** to guarantee immutability and resistance to tampering.

3. **Verified Issuer Identities:** Organizations issuing **WRCode** automations must undergo identity verification conducted through **wrcode.org**. The verified identity is then cryptographically hashed and anchored on the **IOTA DLT**, providing tamper-proof verification of the issuer while safeguarding sensitive identity details. This ensures that only authorized publishers can issue trusted automations and that the authenticity of the issuer can always be validated.

4. **Secure Vault Software:** Vault applications responsible for handling sensitive data must:

- Adhere strictly to **wrcode.org** guidelines.
- Be published under the **GNU Affero General Public License v3 (AGPLv3)** , the same license used by the OpenGiraffe project.
- Be cryptographically hashed.
- Be anchored on the **IOTA blockchain** to ensure authenticity, traceability, and tamper-proof verification.

The Vault must operate as a **standalone, independent application** to guarantee that all sensitive actions, including PII handling and automation approvals, occur within an untampered, auditable environment. All **whitelisting of WRCode providers** and **user consent decisions** must be made exclusively within this original Vault application, ensuring that no proprietary software can override or simulate user approval.

To prevent unauthorized duplication or misuse, each Vault instance must be **cryptographically bound to the device on which it is installed** . This is achieved by generating a **device-specific hardware fingerprint** derived from secure elements such as a Trusted Platform Module (TPM), Secure Enclave, or equivalent. The hardware fingerprint, combined with the Vault software hash, is **anchored on the IOTA blockchain** to establish a tamper-proof link between the Vault and the device.

At runtime, the Vault must verify its own software integrity and confirm the device's hardware fingerprint before permitting any sensitive operation or data access. This ensures that even if Vault software is copied or tampered with, PII remains inaccessible without the original, trusted hardware environment.

To maintain frictionless user experiences for trusted services, the Vault must allow users to **explicitly approve initial PII access** and optionally **whitelist specific WRCode providers** for streamlined future interactions. This whitelisting remains entirely under user control, with the ability to review, revoke, or modify permissions at any time.

To maintain frictionless user experiences for trusted services, the Vault must allow users to **explicitly approve initial PII access** and optionally **whitelist specific WRCode providers** for streamlined future interactions. This whitelisting remains entirely under user control, with the ability to review, revoke, or modify permissions at any time. Access to PII is only granted if:

1. The proprietary software calling the Vault passes integrity checks.
2. The Vault software matches the original, hashed version anchored on **IOTA**.
3. The user has explicitly granted or whitelisted the specific provider.

Vaults may optionally log user approvals and whitelisting events locally in a tamper-proof way to enhance transparency, user control, or compliance needs. These logs are strictly informational and do not contribute to the core security of the WRCode framework, which is fully enforced through cryptographic proofs, anchored software integrity, and device-bound Vault verification. Proprietary WRCode apps must automatically anchor the hashes of every new software release on IOTA to maintain verifiable software integrity. This ensures that all participants follow the guidelines strictly and that users can trust the process without compromise.

Only Vaults issued by verified WRCode publishers and listed on wrcode.org are considered compliant and safe for use.

Together, these elements establish a comprehensive security model that protects users from fraud, manipulation, and unauthorized data access.

Unregistered automations are strongly discouraged, as they expose users to significant risks, including:

- Use of unverified Vault software that may mishandle or leak sensitive data.
- Execution of altered or insecure automation templates.
- Absence of verifiable issuer identity or template integrity.

Only **WRCode**-compliant automations—fully registered, verifiable, and tamper-proof—should be trusted for customer-facing or security-sensitive applications.

Consumers and enterprises must:

- Use only **Vaults published by verified WRCode issuers** listed on **wrcode.org**.
- Verify automation templates through the public registry before execution.

To maintain openness and flexibility, any organization may issue its own Vault software, provided that the software is **fully open-source, published on wrcode.org, auditable, and anchored tamper-proof on IOTA**.

Such Vault applications must:

- Be published under an approved open-source license.
- Follow the strict security and privacy guidelines set by **wrcode.org**.
- Be cryptographically hashed and anchored on **IOTA** to ensure authenticity, integrity, and verifiability.

This approach allows companies to build their own compliant Vault solutions while maintaining the core trust and security guarantees of the **WRCode** ecosystem.

This framework prevents fraudulent actors from manipulating automations or distributing unauthorized Vaults.

Future Execution Layer: **QRGiraffe**

The forthcoming **QRGiraffe** orchestration environment will leverage the full potential of the **OpenGiraffe AI Agent Orchestrator** to extend and enhance the capabilities of WRCode-based automations. QRGiraffe will not be required for WRCode adoption but will serve as an example of how to build advanced, AI-driven automation experiences on top of the WRCode standard.

Timestamped, Open Innovation

The **WRCode standard** has been:

- **Publicly documented and cryptographically timestamped** using **OpenTimestamps on the Bitcoin blockchain** .
- Released under an open, community-driven model to ensure transparency and verifiability.

This guarantees the authenticity, originality, and trustworthiness of the **WRCode** framework.

The Importance of **WRCode**

Without a secure standard, QR-based automations would remain vulnerable to:

- Tampering and manipulation
- Spoofed automations and phishing
- Lack of issuer accountability
- Privacy breaches

The **WRCode** standard addresses these risks through:

- Verified, tamper-proof automation triggers
- Transparent validation and accountability
- Strict Vault and privacy safeguards

Conclusion

The WRCode establishes a **trusted, transparent, and verifiable global standard** for QR-triggered automation:

- Only QR Codes that meet **wrcode.org** standards are recognized as **WRCode**.
- All public **WRCode** automations must be **registered, anchored, and publicly verifiable** .
- Vault software must be **secured, verifiable, and anchored on IOTA** .

By following these principles, **WRCode** empowers organizations and individuals to adopt QR-triggered automation in a manner that is secure, accountable, and privacy-respecting.

Hotel Check-In Using **WRCode**: Practical Example

Scenario Overview

A guest arrives at a hotel without a prior booking. In the lobby, a **digital screen displays a dynamically generated WRCode** that includes all relevant booking details, such as room availability, pricing, and optional services. The guest scans the **WRCode** using their smartphone or another device equipped with a **WRCode**-compliant Vault.

This touchless process allows the guest to instantly review the booking information directly on their device. With a single secure confirmation in the Vault, the booking, payment, and check-in process is executed automatically. The **WRCode** ensures that all automations, including PII handling and transaction processing, are completed securely and transparently without the need to interact with hotel staff.

Step 1: Immediate Check-In

- The **WRCode** triggers a PII consent screen in the Vault.

- The guest approves, and the Vault validates the **WRCode** through **zero-knowledge proof online verification** (hash-only lookup on IOTA).
- The guest's preferences—such as room temperature, pillow type, or loyalty rewards—are preconfigured by the guest using the hotel's **WRCode** generator available on the hotel's website. Through fine-grained slider settings and service choices, the guest creates a personalized **WRCode** in advance, which is securely stored in the Vault as non-sensitive external data specifically for later use in automated booking and check-in.

Step 2: Arrival & Access

- Upon arrival at the hotel room, the guest uses the same **WRCode** to unlock the door.
- All preferences are pre-applied without additional steps.
- The Vault ensures that no sensitive data is shared unless validated and approved.

Step 3: Hotel Services and Stay

- The guest can use additional **WRCode**-enabled services:
  - Order drinks or food by scanning **WR Codes** in the room or bar, with charges linked to the room or paid directly via a secure Vault-stored payment method.

    - Book spa appointments or restaurant tables by scanning other WR Codes around the hotel.
    - Connect to the OpenGiraffe Orchestrator on a laptop or tablet to access personalized suggestions: upcoming events, late checkout offers, or tailored experiences.
- The AI automation templates embedded in the WRCode are executed automatically to optimize the stay.
- An intelligent OpenGiraffe optimization layer runs in parallel, checking for better options or dynamic recommendations based on the guest's preferences, unless toggled off.

Step 4: Check-Out

- Check-out is fully automated via WRCode or the orchestrator.
- Guests receive digital invoices, loyalty point updates, and follow-up service offers.

Security and Privacy

- All WRCode automations are executed only after zero-knowledge proof validation using IOTA anchors.
- No sensitive PII is ever shared or processed without full consent.

- Unverified WR Codes are discarded automatically without execution.
- Optional transaction logs for orders or service confirmations are stored securely within the Vault for verifiability but without exposing personal details.

Offline Use

- If scanned offline, a **WRCode** behaves like a normal QR Code, displaying basic non-sensitive information.
- If a **WRCode** is detected but cannot be verified immediately, it is securely stored separately from PII until validation is possible. If it fails validation, it is deleted without execution.

TRUSTLESS
SECURITY
with
WRCode

WRCode represents a proposed standard for secure, verifiable QR Code-triggered automation. Building on familiar QR Code technology, **WRCode** introduces strict security guidelines, account-based identity anchoring, and trustless user control while maintaining accessibility and privacy. This paper outlines additional security, resilience, and automation concepts that complete the **WRCode** ecosystem.

Core Principles

- Mandatory Identity for **WRCode** Issuers: All entities generating **WRCode** automations—including merchants, service providers, and organizations—must register through wrcode.org and undergo identity verification. Their identities are anchored tamper-proof on the IOTA blockchain as part of **WRCode's** foundational security design.

- Free Access for Users: Individual users interacting with **WRCode** automations retain the freedom to do so without mandatory identification. Identity anchoring for users is strictly optional and only applies where specific use cases (e.g., regulatory compliance, service personalization, or recovery) require it.

Optional Identity Anchoring for Users

Users may opt into a lightweight identity anchoring system to:

- Unlock faster service in high-trust scenarios (e.g., hotel check-in, car rental).

- Enable automated invoicing, business travel expense management, and personalization.
- Facilitate optional Vault recovery and revocation in case of device loss, ensuring users can securely disable lost Vaults and restore their identity on a new device.

The identity anchor is derived exclusively from the user’s email address, cryptographically hashed and securely anchored on IOTA. No other personal information is collected, exposed, or shared with WRCode providers, and the email remains private. It is neither published nor shared unless required by lawful authority. WRCode.org’s role is limited to verifying the authenticity and integrity of the anchor without the ability to access or disclose the user’s identity.

Resilience: Trustless Vault Backup and Revocation

To ensure continuity and user safety in case of device loss, WRCode introduces an optional backup and recovery feature based on trustless security by design:

- Encrypted Backup: Vaults are backed up locally or optionally in encrypted cloud storage, always tied to the user’s WRCode account (email address). A cryptographic hash of the original device’s hardware fingerprint must be saved by the user (on a USB stick, hardware key, or secure medium). Restoration requires:

    1. The verified email account.
    2. The hardware fingerprint from the original device.

3. The user-defined passphrase.

- Mandatory Multi-Factor Recovery: A portion of the recovery data is stored as an account-bound hash fragment. Without all three factors—account access, device fingerprint, and passphrase—the Vault remains permanently inaccessible. Even if the Vault container is copied, it cannot be decrypted without the full set of credentials.
- Tamper-Proof Revocation: In case of device loss, users can revoke the old Vault and issue a new Vault bound to new hardware, ensuring continuity without compromising security. This recovery feature is offered as part of WRCode's optional premium service tier.
- Self-Controlled: Neither WRCode.org nor any third party holds decryption capability. All sensitive operations remain exclusively under user control.

This layered approach ensures that Vaults cannot be stolen, copied, or misused—upholding true trustless security.

Contextual Automation: Local AI Assistance

To enhance usability while preserving privacy, WRCode supports the use of lightweight, device-local language models (LLMs), securely stored and executed within the Vault:

- Users can issue simple voice or text commands (e.g., “Late checkout” or “Order coffee”) processed entirely on-device.

- The local LLM assists in mapping user preferences to automation templates without exposing data to external servers.

Paperless Payments. Automated Outcomes.
WRCode
WRCode
Printing receipts isn’t good for the environment. Thermal paper often uses chemicals that put people and the environment at risk.

Scenario: Checkout Reinvented—Private, Paperless, Automated

Lisa visits her favorite fashion store. Instead of receiving a printed receipt, the checkout screen displays a WRCode—a secure, tamper-proof QR Code linked to a verifiable automation template.

Step 1: Scan, Pay, and Share Preferences in One Flow

- Lisa scans the WRCode using her WRCode-compatible app.
- The app:
    - Retrieves Lisa's secure identity anchor (a cryptographically hashed zero-knowledge proof anchor stored on IOTA).
    - Uses her PII Vault, where her preferred payment method (e.g., credit card, Apple Pay, crypto wallet, QR Pay or Cash) is already securely stored.
    - Executes the payment directly from the Vault—no need to re-enter any sensitive information. Simple confirmation per fingerprint or FaceID.

Lisa's payment credentials are securely provided to the WRCode-verified merchant and the transaction is logged tamper-proof on IOTA for future audits.

The store sees only the payment data necessary for transaction execution—nothing more.

Step 2: Automated Receipt & Personalization (Optional)

- Immediately after checkout, a second WRCode appears.
- This **WRCode** is:
    - Dynamically tailored to Lisa's pre-shared preferences (e.g., she uses Lexoffice for taxes).
    - Securely linked to the transaction and ready to trigger automation.

Lisa skips scanning this second code in the shop—it's optional.

Step 3: Deferred Automation via Orchestrator

- At home, Lisa opens her Orchestrator App.
- The app:
    - Recognizes her WRTransaction, stored securely in the Vault alongside the payment proof (no paper, no email).
    - Displays the personalized second **WRCode** linked to her earlier payment.
- Lisa scans this **WRCode**—because she had securely shared some basic preferences earlier from her Vault, the merchant had already dynamically tailored this second **WRCode** during checkout and attached it to the transaction:

    - The invoice is enhanced with her Vault PII (e.g., name, address, tax number).
    - Automatically submitted to Lexoffice, WISO Tax, Taxman or other integrated systems based on the taylored automation template.
    - The entire process is anchored on IOTA—verifiable, timestamped, and tamper-proof.

Key Innovation: Payment Data Stored and Controlled in the Vault

- Users store multiple payment options in the PII Vault:
    - Credit/Debit Cards (tokenized or via secure keychain)
    - QR Pay (e.g., Alipay, WeChat Pay, SEPA QR, Girocode, PayPal QR)
    - Crypto wallets (optional future integration)
- The payment is initiated via the Vault but—for payment execution—data is securely shared with the WRCode-verified merchant.
- The transaction details are immutably logged on IOTA, ensuring legal defensibility, privacy, and compliance.

- Even contactless in-store payments can be triggered by **WRCode** → Vault → Payment in seconds.

Strategic Messaging: WRCode as the Missing Link Between Payment, Automation, and Privacy

Tagline Ideas:

1. "One Scan. One Pay. One Click to Automate Your Life."
2. "Receipts That Write Themselves. Taxes That File Themselves."
3. "No Paper. No Apps. Just You, Your Vault, and the Power to Automate."

Environmental Impact: Why Paperless Matters

- Thermal Paper Receipts Are Harmful: Most printed receipts use thermal paper coated with chemicals such as BPA or BPS, which are known to be harmful to both human health and the environment.

- Environmental Waste: Billions of paper receipts are printed every year, generating unnecessary waste and contributing to deforestation, pollution, and landfill overflow.
- WRCode Eliminates the Need for Printed Receipts: By moving to digital, verifiable receipts via WRCode, merchants and consumers help reduce chemical exposure, save trees, and cut waste.

Bonus Features to Market:

- Budget Automation: WRCode preferences could link expenses directly to personal budgets or savings plans.
- No App Lock-In: The open standard allows multiple wallet and vault apps to co-exist.
- True Privacy: Payment data is shared only with verified merchants and recorded immutably for compliance—never with third parties, only with WRCode-verified WRCode providers as necessary for the automation.

This is not just payment automation—it's identity + payment + compliance + automation in one frictionless moment.

Introducing WRCode: A Trustless Automation Framework for Payment, Identity, and Loyalty

The digital landscape is evolving rapidly, and businesses require flexible, secure, and privacy-first solutions that adapt to a variety of use cases. **WRCode** represents a next-generation framework that enables merchants, service providers, and users to interact seamlessly through decentralized automation. Unlike traditional systems that rely on centralized databases, **WRCode** offers a modular, trustless approach that gives control back to users while enhancing business opportunities.

WRCode is free to use for all end users, including those who wish to benefit from paperless receipts, rewards, and basic bonuses—without the need for any registration. Some advanced features—such as quick onboarding, fast-lane check-ins, formal invoicing with required business information, or multi-device vault recovery—may require an optional lightweight registration. This registration ensures that both compliance and advanced convenience features can be offered while maintaining **WRCode's** core principle of user control and privacy-first design.

The Core: WRCode.org

At the heart of the system lies **WRCode.org** — the open protocol that powers all WR-branded modules. It serves as the secure, standardized foundation that ensures interoperability, privacy, and flexibility across various industries.

Key Modules Explained

WRPay — Privacy-First Payment

WRPay enables fast, flexible payment solutions that respect user privacy. Whether used for digital transactions or as a paperless complement to cash payments, WRPay eliminates unnecessary data exposure while offering merchants new automation possibilities.

Benefits over traditional payment systems:

- Only absolutely necessary data shared
- Paperless receipts, eco-friendly
- Optional backend automation triggers for tailored experiences

WRCollect — Loyalty & Rewards Reinvented

WRCollect allows businesses to offer dynamic, customizable loyalty programs without intrusive data collection. Customers "collect" benefits, such as discounts, gifts, or exclusive offers, by interacting with WR-enabled merchants. Unlike traditional systems that centralize loyalty data, WRCollect stores information locally or encrypted, keeping user preferences private.

For merchants, **WRCollect** provides a powerful incentive tool: it enables creative and personalized rewards that go beyond simple points or generic discounts. For example, a customer who spends over a certain amount might receive a free coffee on their next visit or unlock exclusive deals—entirely automated through **WRCode** without complex backend systems. This could mean: "Collect a bottle of wine on your next shopping trip" or "Unlock a personalized discount after your third visit," with all interactions remaining privacy-friendly and seamless.

Benefits over traditional loyalty systems:

- No central data mining
- Interoperable across merchants
- Customizable benefits and creative incentives that drive repeat business
- Only absolutely necessary data shared
- Paperless receipts, eco-friendly
- Optional backend automation triggers for tailored experiences

WRPass — Decentralized Identity & Access

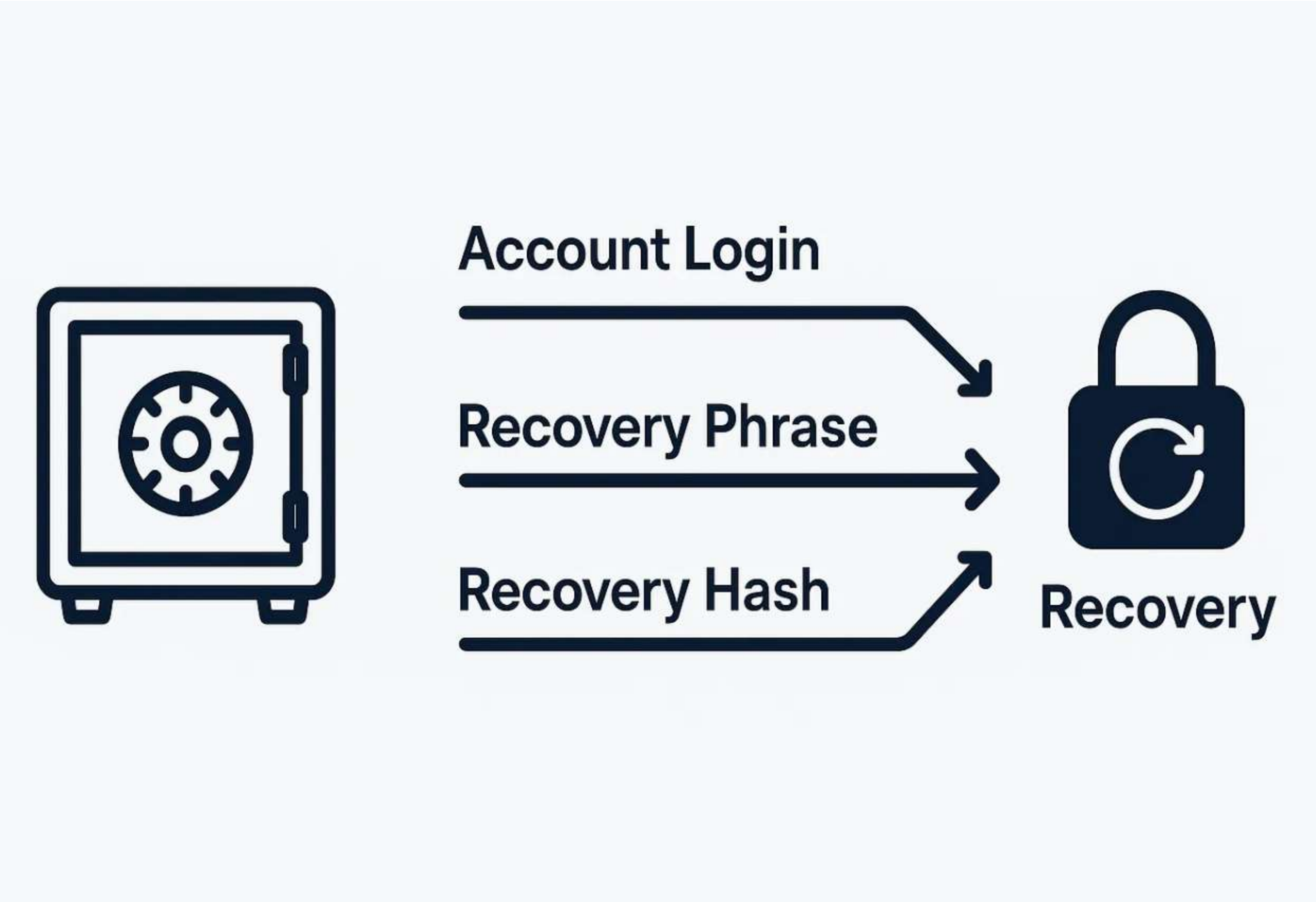
WRPass replaces conventional logins and verification methods with a secure, WR-based identity layer. Acting as an on-premise, **open-source** password manager and verification tool, **WRPass** enables all types of identity proofs—whether zero-knowledge-based or traditional—depending on the specific use case. Users retain full control over their credentials through a local vault, allowing trustless, secure verification for both digital and physical access while ensuring sensitive data never leaves their device unless explicitly shared.

Benefits over traditional identity systems:

- No passwords stored in the cloud
- Verifications without revealing unnecessary information
- Suitable for digital and physical access control

4. WRVault — Secure Data Storage

WRVault serves as a private, user-controlled storage space for sensitive data, preferences, or automation keys. Each vault is cryptographically bound to the user's **WRCode** account, ensuring that only the rightful account holder can manage, revoke, or migrate vaults. Upon creation, users generate a unique recovery hash, which is stored securely offline (e.g., on a USB stick or printed backup). This recovery hash, combined with account access, is required to trigger any vault revocation or migration process.


Account Login
Recovery Phrase
Recovery Hash
Recovery

To maintain strict security, WRCode.org does not store recovery hashes, passphrases, or vault keys. Only a non-reversible public fingerprint of the vault is optionally stored to allow users to reference and manage their vaults securely. Revocation or migration requests must be authorized both through account login and the presentation of the user-held recovery proof, ensuring that no attacker or insider can trigger revocations without the user's active involvement.

If both the recovery hash and any optional passphrase are lost, the vault becomes unrecoverable by design—ensuring true zero-trust security without any central killswitch. Even in the unlikely event of a WRCode.org server breach, attackers cannot mass-revoke or access vaults (vaults are store locally anyway), as critical recovery materials are never centrally stored and remain fully under user control.

For advanced users who wish to benefit from more powerful cloud-based AI services, WRVault offers configurable options where users can explicitly authorize selected data to be processed externally. This flexibility preserves the core principles of trustless design and full user control, while allowing tailored experiences for those who opt in.

Both the original structured vault data and any embedded semantic layer coexist on-premise or within a user-selected architecture, maintaining security while unlocking the power of AI-enhanced workflows.

Compliance and Invoicing in WRCode

WRCode enables fully paperless, anonymous invoicing for everyday purchases by default. In most cases, this is legally sufficient and respects the user's privacy.

When merchants need to issue formal invoices that include regulated data (such as tax numbers or business details), the WR application they use must include the necessary business logic to prompt users for optional data disclosure.

Users can choose to:

- Share a zero-knowledge proof identity (email + token) to receive compliant digital invoices.
- Share unmasked personal information (PII) if required for bookkeeping or legal reasons.
- Or receive an anonymous invoice and add necessary details manually later if they wish to claim business expenses or tax benefits.

The control remains entirely with the user, while merchants retain the ability to meet regulatory needs within their own WR applications.

🤖 OpenGiraffe: Real-Time AI Orchestration for Mobile, AR, Robots, and Desktop—With Predictive, Privacy-Preserving Automation

OpenGiraffe is a modular AI orchestration framework that delivers real-time, context-driven automation across desktop workstations, mobile devices, AR headsets, and even robots—while maintaining privacy, human oversight, and local data control.

At its core is the OpenGiraffe Orchestrator, a desktop-based control layer that holds the company's entire knowledge base—including technical documentation, compliance checklists, and best practices—ensuring that every user, device, or agent receives consistent, expert-level guidance.

How It Works:

1. The **orchestrator** runs on secure local infrastructure, coordinating AI-driven workflows based on:
   - **User role**
   - **Global task intent and ai agent logic (predefined by templates)**
   - **Real-time contextual inputs** (e.g., geolocation, ZKP-verifications, environmental signals, sensor data, vision data, audio data and other input data)
2. Outputs are streamed to:

  - 📺 Desktop displays / VR Devices
  - 📱 Mobile devices
  - 🥽 AR overlays
  - 🤖 Robots or smart devices

The backend triggers AI-powered workflows in real time:

- 🧠 An LLM agent interprets the user's role, context and task history
- 🎯 Predefined workflows or generated workflows are selected
- 📱 Results are streamed back to the user's smartphone or AR overlay as visual information, task lists, alerts, or step-by-step instructions
- 📼 Optionally, a session can include local video/audio capture , stored securely for documentation or post-processing—entirely within user-owned infrastructure
- The AR device or smartphone effectively acts as the **mobile master interface**, continuously collecting contextual input—including geofence events, GPS, user role, environmental data, speech commands, or text interactions. This input is securely looped to a **local or edge-based OpenGiraffe orchestrator**, which interprets the intent and triggers predefined or dynamically generated automation workflows. The user does not need to manually initiate processes;

instead, **presence and context alone** can activate backend logic. Optional **voice or text commands**—such as *“Show safety checklist”* or *“Draft report for Zone 3”*—can further refine or redirect the orchestration. Confirmations, overlays, or task results are streamed back in real time to the user’s AR display or smartphone, enabling fully automated, hands-free interaction **without reliance on cloud infrastructure or invasive tracking**.

Example: Geolocation + Vision: Predictive, Context-Aware Support

- By combining **geolocation data** (e.g., GPS, geofence) with **visual inputs** (from cameras, AR devices, or sensors), the orchestrator can:
    - Accurately **understand the technician’s environment** and **current task**.
    - **Predict likely next steps** or **possible risks and the most efficient approaches to fulfill the task**.
    - **Time the delivery of guidance** precisely—showing checklists, warnings, or support **exactly when needed**, without overwhelming the user with constant notifications. (AI Agents can adjust it dynamically)
- For example:
    - A technician arrives at a site (**GPS triggers intent recognition, ZKF verified if needed**).
    - The AR glasses detect a specific machine type (**vision input**).

  - The orchestrator **automatically surfaces relevant repair steps, known issues, or safety warnings**—without being explicitly asked.
- This **proactive assistance** is:
  - ✅ Context-sensitive
  - ✅ Privacy-preserving (no personal data leaves the infrastructure)
  - ✅ Fully **overridable** or **tunable** via **voice commands** or manual control.

Human, Robot, and Machine Collaboration:

- **AR devices, smartphones, and robots** can all act as **master tabs**—initiating or receiving tasks based on environmental detection or operator commands.
- Robots can contribute real-time data (e.g., location, vision, task state), helping the orchestrator to:
  - **Predict needs before they arise**
  - **Trigger the right AI-driven workflow at the optimal moment**
  - **Loop in human supervisors when ambiguity or exceptions occur**

Key Advantages:

✅ **Centralized Knowledge Repository:**
All company knowledge is stored **locally**—keeping data safe and ensuring reliable, standardized support across the entire operation.

✅ **Predictive, Just-in-Time Automation:**
By combining **geolocation** and **vision data**, the orchestrator can **anticipate task needs** and provide support **at the right time—without unnecessary prompts**.

✅ **Device-Independent Orchestration:**
Desktops, mobile devices, AR glasses, and **robots** all work together in a **coordinated, flexible automation system**.

✅ **Privacy and Compliance by Design:**
Sensitive information never leaves the local environment. Optional **Zero-Knowledge Proofs** and **IOTA anchoring** ensure **verifiability without exposure**.

✅ **Human-Centric:**
Automation assists—but never replaces—humans. The operator stays in control, with **support that can be refined, paused, or overridden at any moment**.

In short: **OpenGiraffe** enables **context-driven, predictive AI orchestration** across people, devices, and machines—delivering **the right help at the right moment, powered by local knowledge, secured by design, and always under human control**. All actions can be logged locally , and the orchestrator can sync anonymized metadata or Zero-Knowledge-Proofs to a central server if needed—but **raw data and personal context never leave the edge network** .

🔐 Core Principles of OpenGiraffe Geofence Automation (AR/Mobile Mode)

- ✅ **Local ZKP Generation** – Presence is proven, but not revealed
- 📡 **Edge-Synced AI Logic** – Smartphones and AR glasses connect to a secure local orchestrator
- 🧠 **Intent-Aware Workflows** – Role-, time-, and task-based logic selects appropriate responses
- 🧍 **No Tabs, No Dashboards** – UI is adapted for compact screens and overlays, local orchestrator runs the ai orchestration optionally guided by a backend operator
- 📲 **Streaming Outputs** – Real-time instructions, summaries, or media rendered on mobile/AR interfaces
- 🔐 **Private Recording** – Sessions can be optionally logged (video/audio) under full local control
- 🛠️ **Composable & Offline-Ready** – No dependency on public cloud or internet; logic runs on-site
- 📜 **Verifiable, Auditable, Explainable** – Every input, activity, reasoning step, or geolocation event can be cryptographically anchored—optionally on a blockchain like IOTA—and remains fully user-auditable. This ensures transparent, tamper-proof records without exposing personal data,

enabling organizations to verify actions, context, and decisions at any time.This architecture makes **OpenGiraffe** ideally suited for **in-the-field AI interaction** , **hands-free workflows**, and **privacy-first automation** across mobile and wearable form factors—whether in labs, warehouses, medical settings, or decentralized field operations.

Shops and zones can be whitelisted in advance, and geolocation-based detection can be toggled on or off at any time. Since only anonymous cryptographic hashes are generated locally not even at the vendor site—without storing or sharing personal data—users remain fully in control and can decide when and where presence proofs are created.

🔄 Generalization Across Trigger Modalities

While this section focused on geofencing, the underlying principles of **OpenGiraffe**—namely, **zero-knowledge validation of contextual triggers** , combined with **intent-aware AI orchestration** —are **technology-agnostic** . The same local verification and backend automation flow can be extended to a wide range of context-detection methods:

- **BLE proximity beacons**
- **NFC or WR Code scans**
- **Wi-Fi SSID presence**

- Camera-based zone detection
- Voice or gesture commands

Each of these modalities can serve as a secure, local trigger—optionally wrapped in a ZKP claim—enabling **OpenGiraffe** to offer **privacy-preserving, explainable, real-time automation** across diverse environments such as industrial zones, smart offices, AR workspaces, or mobile field operations.

🚨 Just Another Possible Scenario: AI-Supported Alarm Systems

Using **Optimando's** architecture, users could configure a security setup where motion detectors trigger a live video stream analyzed by AI—capable of recognizing covered faces or visible weapons. In such cases, the system could escalate the incident, potentially even alerting authorities nearly in real-time. This scenario illustrates how **OpenGiraffe** empowers users to orchestrate advanced AI workflows—extending far beyond the use cases covered in this paper.

🖥️ Technical Considerations and Limitations

Some connected services may block automation, especially if login sessions expire or DOM manipulation is restricted. In such cases, the system provides fallbacks:

- Users manually log into relevant services in browser tabs
- Agents use visual or metadata-based cues to navigate and trigger content display
- Inaccessible features are flagged, and alternatives (e.g., via n8n, email relay, or webhooks) are proposed

The OpenGiraffe architecture ensures reliability while retaining flexibility.

⚙ Optimized for High-Performance Workstations with Scalable Backend Flexibility

Optimando.ai is engineered to take full advantage of modern high-performance workstations — particularly those equipped with multi-core CPUs and high-end GPUs such as the NVIDIA RTX 5090. This setup supports fast, local execution of AI processes with high parallel throughput and minimal latency. It is especially well-suited for the demands of real-time, browser-based multi-agent orchestration.

📋 Local Performance Capabilities

• Local LLM Acceleration

High-end GPUs enable efficient local inference of large language models such as LLaMA 3 or Mistral. A 5090-class GPU can process complex prompts with high token throughput and low latency, which is essential in multi-agent workflows where multiple reasoning paths run concurrently and interdependently.

- Real-Time Speech-to-Text

Transcription engines like Whisper (Large) benefit from GPU acceleration, enabling continuous, high-fidelity speech input to be processed in parallel with other tasks — critical for multimodal input scenarios and hands-free interactions.

- Parallel Agent Execution
  With sufficient VRAM and compute power, multiple AI agents can operate simultaneously without affecting system responsiveness. This is particularly valuable in environments with multiple display slots, where agents generate, update, and refine results asynchronously.

🧩 Optional Automation Layer via n8n

While Optimando.ai is fully functional on its own, it can optionally be extended by integrating n8n as a backend automation engine — particularly useful for advanced or multi-layered workflow requirements. n8n is a separate system, but its modular, browser-based, and extensible design complements Optimando's orchestration principle perfectly.

When used together:

- OpenGiraffe handles frontend orchestration and visualization of agent outputs,
- while n8n manages backend automation tasks such as:
  - multi-step validations,

  - scheduling,
  - agent chaining,
  - file processing,
  - external API interactions.

On powerful local hardware, n8n can run side-by-side with Optimando, offering low-latency automation while keeping full control over data flow and system logic.

🌐 Scalable Design for All Hardware Classes

For less powerful systems — such as laptops or entry-level desktops — n8n can be hosted on a dedicated self-hosted server. In this configuration, frontend devices act purely as lightweight visualization interfaces: they receive and render the output of AI agents without executing any heavy computation locally.

This architectural separation allows even modest hardware to participate in complex, distributed AI workflows without performance issues.

Importantly, all core components in this stack can be self-hosted and are either open source or source-available:

- Optimando.ai (frontend orchestration layer),

- local LLM backends (e.g., via Ollama),
- vector databases (e.g., Qdrant, Postgres),
- and the automation layer (e.g., n8n).

While n8n is not open source in the OSI sense, it is licensed under a *source-available* model (Sustainable Use License). This means the full source code is publicly available, auditable, and modifiable for self-hosted use — preserving transparency and enabling complete control over system behavior in professional environments.

✅ Summary

The OpenGiraffe architecture is capable of scaling from lightweight, local deployments to high-performance multi-agent orchestration environments — all while maintaining user transparency, data sovereignty, and full modularity. When combined with optional backend tools like n8n, it offers a powerful, extensible, and privacy-respecting alternative to centralized SaaS AI platforms.

Multi-Screen and VR Environments

The orchestration interface is built to take advantage of extended display setups and VR headsets. Multi-monitor workstations and VR browsers enable spatial separation of inputs, outputs, and visual control panels.

Skilled User Configuration

Although normal users can operate the system with preconfigured templates, its advanced features are best utilized by professionals with a background in:

- Local deployment of LLMs or AI tools
- Configuration of connector protocols like MCP, ChatGPT connectors, or custom APIs
- Orchestration of automation flows using systems such as n8n
- Understanding of browser-based control environments and DOM interaction constraints

Templates, aliases, and agent roles can be customized, but a working knowledge of automation frameworks and privacy-aware system design is recommended for full control.

Scalable and Modular by Design

The **OpenGiraffe** architecture is inherently modular and scalable:

- **Entry-level users** can rely on hosted LLMs and run helper agents in minimal configurations.
- **Power users** can run multiple master tabs, integrate local and cloud LLMs, define trigger logic, and connect MCP-compatible services.

This flexibility ensures that the orchestration tool grows with its user base—from individuals testing workflows to full teams managing strategic operations.

In short, while accessible to a broad range of users, the system truly shines in professional hands, running on high-performance hardware, and configured by operators with domain-specific expertise in automation, AI integration, and real-time orchestration.

⚙️ 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.

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 interfaces. 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.

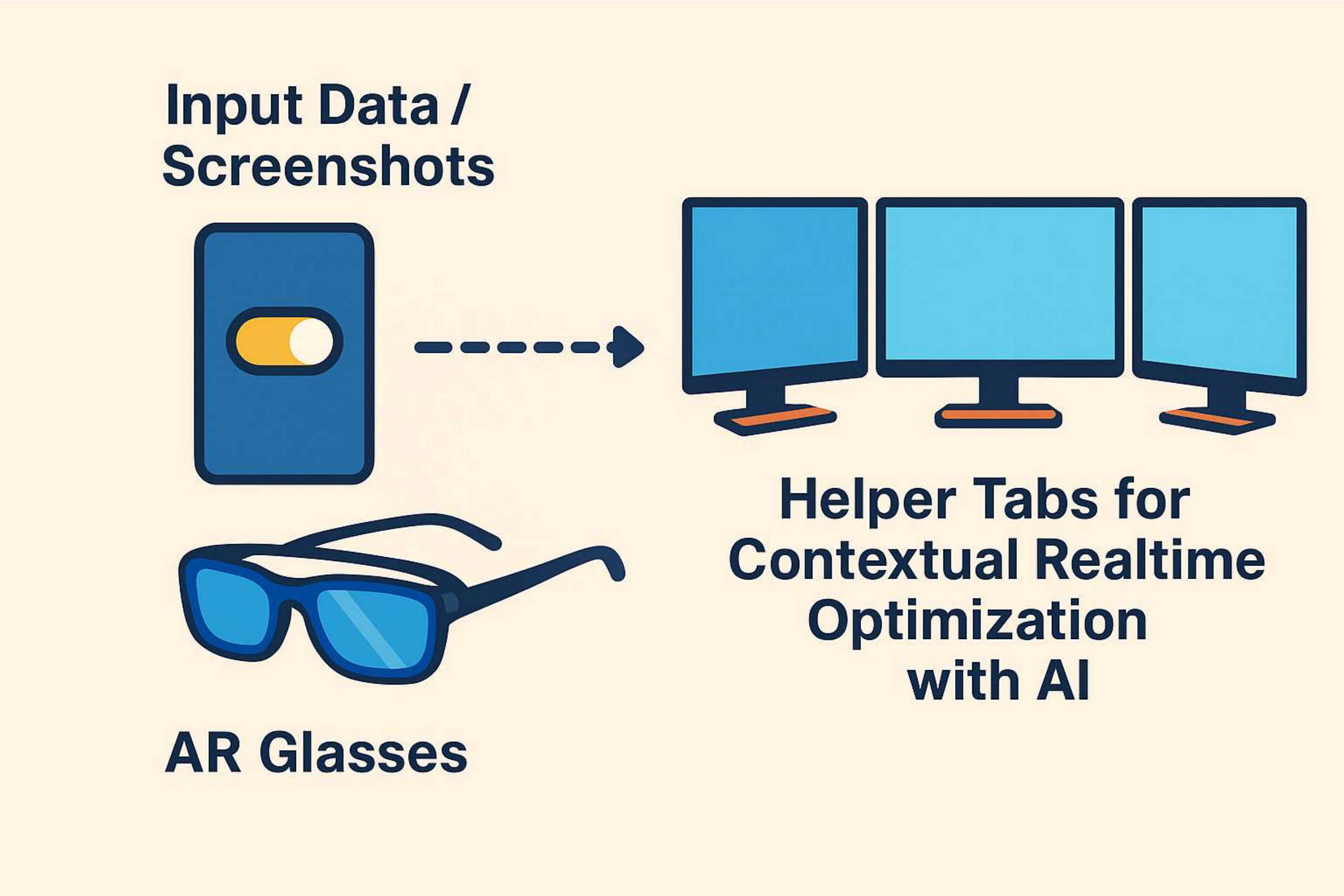
Input data to this system can be **multimodal**, including text, audio, screenshots, UI events, or camera streams. These triggers can originate from within the desktop browser or be activated remotely via smartphones or AR devices acting as **remote master agents**. Once toggled active, such devices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

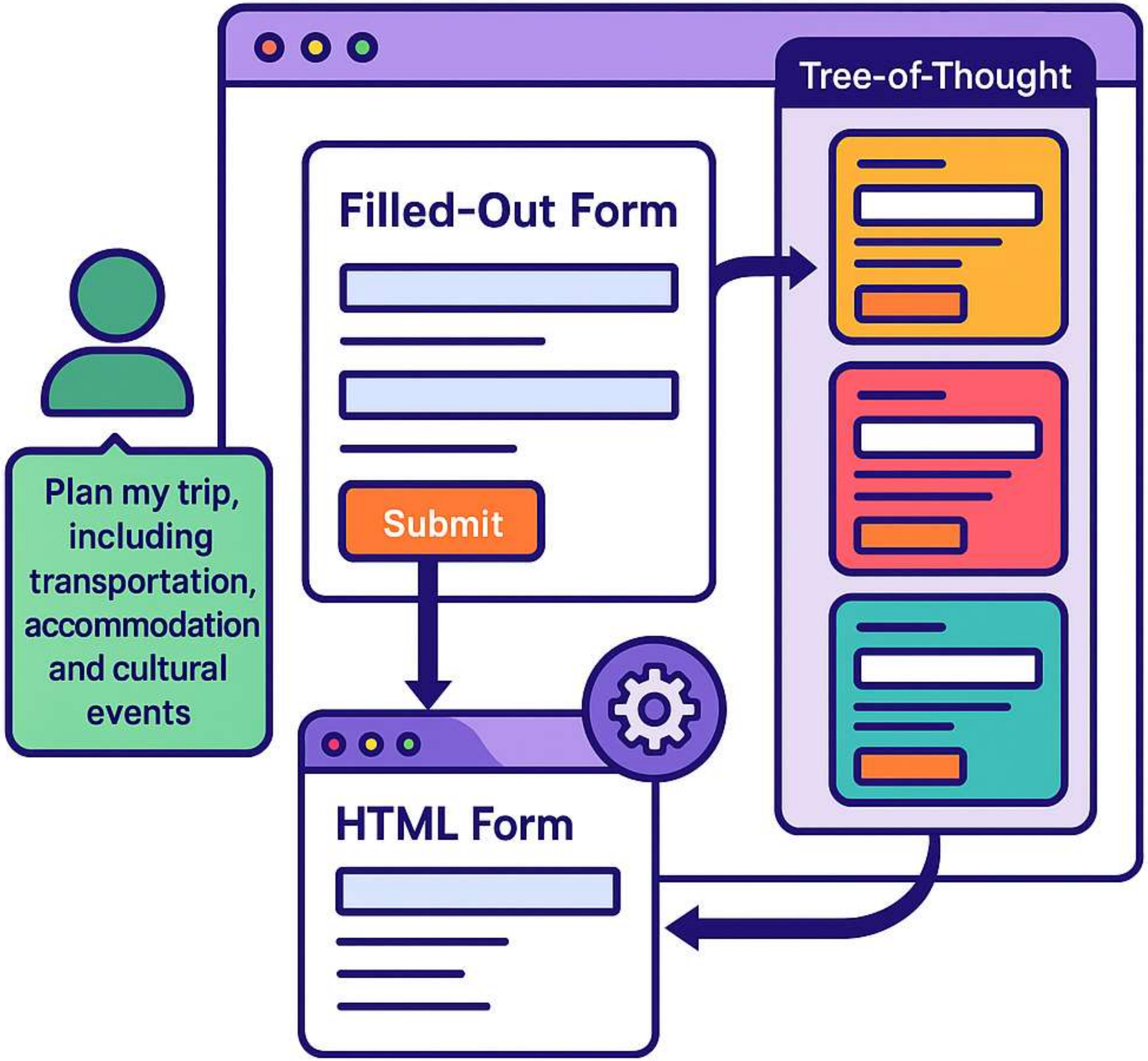
While remote input streams (e.g., from smartphones or AR devices) may transmit data over the internet to the orchestrator, all transmission paths and endpoints are defined and controlled by the user. Optimando.ai **does not operate or provide any backend services** that receive, log, or process this 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 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.


Plan my trip, including transportation, accommodation and cultural events
Filled-Out Form
Submit
Tree-of-Thought
HTML Form

🔧 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**.

🧪 Generalized Example: Cross-Platform Action Planning

A user enters a request via the master input interface:

“Please plan my trip, including transportation, accommodation, and cultural events.”

This triggers a coordinated, multi-agent response:

- A helper agent decomposes the task and distributes subtasks across several pre-defined services (e.g., airline booking portals, hotel sites, ticket platforms).
- Security-first logic ensures that no untrusted website is opened without the user's explicit domain whitelist. If broader exploration is needed, sandboxed environments like Google Mariner may be used — though this introduces profiling considerations.
- User interaction is required for all sensitive operations. The system will not perform actions on its own. Tabs must be opened by the user (e.g., to authenticate into Gmail or a booking portal), or previously opened, authenticated sessions are reused.

Once results are gathered, the best-matching itinerary is synthesized and displayed visually across dedicated output slots — rather than a single frame. This leverages the orchestration system’s adaptive layout, which utilizes available screen space intelligently, in contrast to conventional systems like Google Mariner that condense content into one unified panel.

Each itinerary component is annotated with metadata (e.g., pricing, provider rating, availability).

The master input tab can be toggled off or minimized to reduce distraction. Users may then focus entirely on the results and interact with the system directly via the display slots — including continuing or expanding the session using Tree-of-Thought expansion.

◉ Interactive Output Features

The top result includes functional, user-controlled elements — which may be customized or even generated contextually by AI — but all actions require explicit user approval.

- Submit – Fully functional button, but only executes upon explicit user interaction. The logic behind this can be customized or extended.
- Tree-of-Thought – Reveals pre-processed reasoning paths and diverse alternatives:
  - Different platforms or service providers
  - Alternative travel schedules
  - Price tiers or bundled offers
- Compare & Combine – Allows hybrid planning (e.g., mix Flight A with Hotel B)
- Explain – Displays the reasoning and decision metrics used to generate the top result
- Augment – Enables real-time refinement (e.g., "add airport transfer", "include insurance")

The orchestration system shows only the primary recommendation by default. Users can dive deeper via expansion tools, ensuring an uncluttered yet powerful interface. This configuration empowers users with intelligent, modular autonomy — decisions are enhanced by AI, but always confirmed by the human.

🧾 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.

🧭 **Integration of Metaverse Interfaces into the Browser-Based Orchestration Framework**— Future Outlook (Optional)

While the orchestration framework is based on browser tabs, AI agents, and configurable templates, the **OpenGiraffe** architecture is inherently extensible to virtual environments—**without requiring deep integration into game engines or metaverse platforms**. In this extended use case, the orchestration continues to run on a conventional computer, with AI agents distributed across browser tabs as defined by configuration files. Certain helper agents—originally designed to operate in the background—can optionally be linked to **visual representations within a 3D environment**, such as NPCs (non-playable characters). These NPCs serve purely as **front-end proxies**; the underlying logic, memory, and decision-making processes remain in the external orchestration system.

For instance, in a virtual shop scenario, a user may interact with digital products and approach a cashier NPC to initiate checkout. The NPC itself does not contain embedded logic; rather, it connects to a designated browser tab acting as a helper agent. This tab interfaces with external systems—such as a shop backend, support knowledge base, or legal compliance service—via automation platforms like **n8n** or **MCP-connected agents**. The orchestration layer interprets the user's intent (e.g. purchasing specific items) and generates structured outputs, such as a purchase summary, legal disclaimers, and pricing details. These are presented in-world via spatial overlays or embedded screens—**visual equivalents of the system's display slots**. The user may confirm or cancel the transaction directly within the metaverse, triggering corresponding real-world actions through the connected orchestration backend.

Beyond service and commerce scenarios, this architecture can also be extended to **orchestrate real-time AI-controlled NPC teams**. In such cases, a user (e.g. a player, moderator, or team leader) can issue instructions that are routed through the orchestration backend, which interprets intent and assigns tasks to corresponding NPC agents based on preconfigured logic. This enables the coordination of **multi-agent NPC behaviors**—for example, managing logistics crews, training groups, or support units in real time—without requiring native AI infrastructure within the 3D environment itself.

This decoupled architecture allows metaverse applications to benefit from **advanced AI orchestration, decision logic, and automation**—while keeping the virtual environment lightweight and modular. By separating interaction from execution, it enables fast, maintainable integration of intelligent workflows into immersive spaces, using the same **tab-based orchestration layer** originally developed for browser-native contexts.

🔗 Autonomous NFTs (aNFTs) — Future Outlook (Optional)

As an optional future direction, Autonomous NFTs (aNFTs) could complement OpenGiraffe's orchestration capabilities by enabling new ways to bridge real-world events with decentralized digital assets and automations:

- Policy Reference: NFTs could reference global or local policies to guide AI-driven decisions and trigger actions based on predefined conditions.
- Proof-of-Action: Real-world events (e.g., QR scans, geofence triggers) could mint or update NFTs as verifiable records, optionally anchored on decentralized systems such as IOTA.
- Autonomous Rewards: NFTs could hold assets, rights, or permissions that are dynamically unlocked through verified actions, providing tamper-proof, decentralized incentives.

To enhance privacy, proofs of action or ownership could be anchored without revealing identities. NFTs themselves could also be hashed and used in zero-knowledge proof (ZKP) systems, allowing users to demonstrate NFT ownership or eligibility without exposing wallet addresses or transaction details.

Importantly, OpenGiraffe's AI-driven automation templates could be hashed and anchored on IOTA to ensure trustless verification of AI automation flows. This would allow third parties to independently verify that specific, immutable rules triggered certain automations—enabling AI-driven processes that are both transparent and verifiable without centralized trust. In addition to the templates, also the executions of AI automations themselves could be hashed and logged, providing an immutable, verifiable chain of evidence that documents what rule triggered what action at what time. This ensures that both the existence and the consistent application of rules can be externally validated. In addition, OpenGiraffe could serve as a backend orchestration layer for metaverse platforms or digital identity systems where NFTs or AI-controlled NPCs play a central role. By combining real-world triggers (WR Codes, geofencing) with virtual world actions (NFT updates, digital rewards, AI-driven responses)—all optionally anchored on decentralized systems—OpenGiraffe could enable seamless, privacy-respecting automation that bridges physical and virtual environments. These ideas illustrate potential future extensions beyond OpenGiraffe's core innovation and would require further research and development to ensure security, privacy, and decentralized enforceability.

🔐 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
Master Tab	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. 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.

Component	Privacy Role
	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.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.
Input/Output 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.

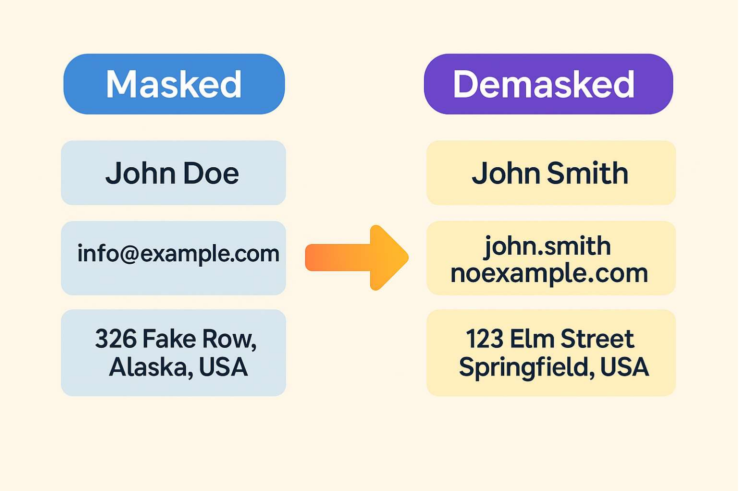
This orchestration strategy allows privacy protection at the execution level, not just via policy.

🛡️ 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)



🧠 Example: Privacy-Preserving Form Automation Using Local LLMs and Embedded Context Memory

When a user visits a structured form — such as a job application, tax return, or official registration — the optimando.ai orchestration framework can detect this intent and assist with intelligent, local form filling.

If the feature is enabled, the system loads the same page in a helper tab, where it attempts to prefill the form automatically. It does so using relevant user data (e.g., resumes, documents, income reports) that are stored locally — optionally within an encrypted container (e.g., VeraCrypt). These files are simply stored on the user's device.

A local LLM (e.g. Mistral) — optionally within an encrypted container (e.g., VeraCrypt), parses the content and embeds it into a private vector database (e.g., Qdrant). From this, it can derive context and field-level data (names, skills, income, etc.) and use DOM manipulation to fill in the form.

The resulting form is shown in a display slot — for example, on a second screen — allowing the user to check, adjust, and submit manually.

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 outputs in display slot 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.

🔐 Local by Default — But Flexible and User-Controlled

The architecture is designed around privacy-first principles, but remains adaptable:

- By default, all tasks involving personal data (PII) are handled exclusively by the local LLM.
- Users retain full control — they decide whether and when to allow external assistance.
- The local model performs all tasks it can handle, especially any involving sensitive data.

This privacy-by-design approach makes it possible to build automation capabilities for white-labeled websites similar to those seen in emerging agentic browser systems — such as Google Project Mariner or Manus — but without exposing profiling-relevant or PII data to third-party services.

☁ Optional Cloud Reasoning — With Controlled Isolation

If the local model cannot solve a more complex reasoning task (e.g. legal interpretation, tax optimization, logic chaining), the user may choose to enable external LLM assistance. In these cases:

- Only non-PII and abstracted fragments are sent to cloud LLMs.
- PII and sensitive identifiers stay on the local machine.
- Profiling-resistant techniques are applied, such as:

    - **Task splitting across providers**
    - **Field-level masking/demasking**
    - **Tree-of-Thought confusion, generating variants to conceal true intent or values**

Additionally, when users directly type information into the master tab (instead of using helper workflows), the system may trigger additional safeguards like DOM-based submit delays or selective field protection.

🛡️ Strategic Privacy Design — Without Overpromising

Optimando.ai is an open-source, evolving framework. Privacy protection is a core design goal — but users should be aware:

- **Not all features are active from day one.**
- **It is built to grow with community input, legal requirements, and open-source innovation.**
- **While it applies state-of-the-art techniques to reduce data exposure and profiling risk, no system can guarantee absolute protection.**
- **The user stays in control, and bears responsibility for how the system is configured and used.**

Unlike many SaaS automation platforms, optimando.ai never assumes full ownership of user data. The goal is to enable powerful AI-assisted workflows — such as "autopilot" for complex digital tasks — without the user needing to hand over sensitive information.

🔐 Optional Encryption – Recommended Best Practice

While optimando.ai does not enforce encryption itself, users can and should take steps to protect their sensitive data. This includes:

- Encrypting files containing personal documents (e.g. resumes, tax PDFs)
- Storing local LLM model files and vector databases inside encrypted containers (e.g. via VeraCrypt or similar tools)
- Ensuring that any local embeddings involving PII or data that can be used for profiling are also secured at rest

Since embedded vectors or fine-tuned LLM memory can contain sensitive content, they should be treated as equivalent to raw data files from a privacy standpoint.

Encryption is entirely optional, but it is strongly recommended — especially for users working on shared machines, mobile devices, or in regulated environments.

This best-practice approach allows users to maintain complete control over their local data environment, without depending on external providers or complex infrastructure.

🔒 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

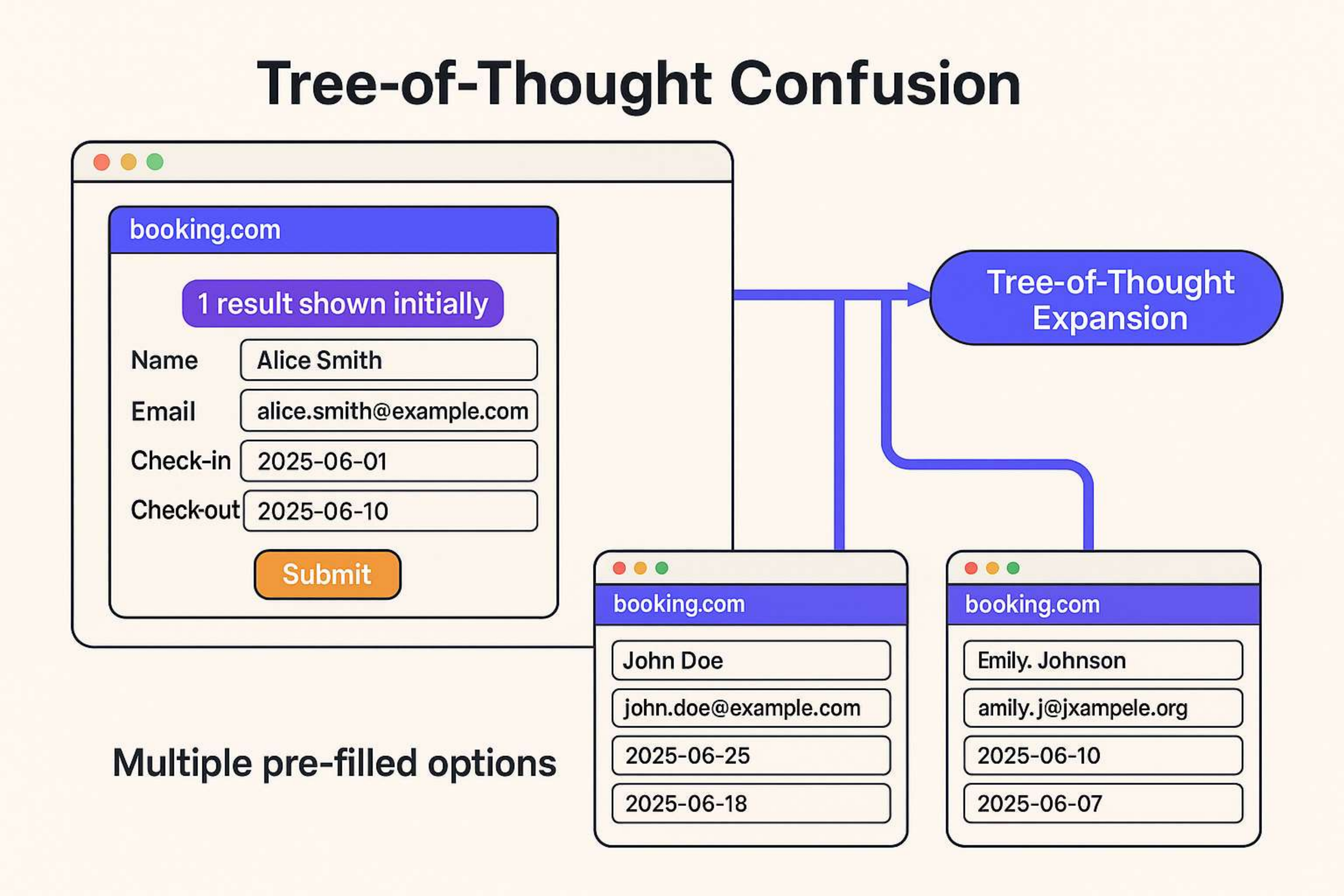
This makes the entire interaction local, transparent, and user-driven.

🌳 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



Tree-of-Thought Confusion

Name Alice Smith
Email alice.smith@example.com
Check-in 2025-06-01
Check-out 2025-06-10

John Doe
john.doe@example.com
2025-06-25
2025-06-18

Emily. Johnson
amily. j@jxampele.org
2025-06-10
2025-06-07

Multiple pre-filled options

Conceptual Feature: Tree-of-Thought Confusion and Profiling Risk Mitigation

A privacy-first feature currently in internal design is the Tree-of-Thought Confusion mechanism, combined with real-time profiling risk detection and dynamic masking/demasking overlays in the Master Tab. It is designed to mitigate unintended profiling when users interact with LLM systems in sensitive contexts such as healthcare, legal, or financial domains.

Example Use Case: “Please find a list of oncologists near Berlin for my ongoing cancer treatment.”

This seemingly routine request reveals highly sensitive personal data. With the growing use of LLMs — across both trusted and opaque platforms — many users remain unaware of the long-term profiling risks. If such profiles are logged, shared, or leaked, the consequences could be significant — from discrimination to surveillance.

This concern is particularly relevant for:

- **Journalists and whistleblowers** , researching or reporting sensitive topics
- **NGO workers or activists** , operating in authoritarian or high-risk environments
- **Professionals and individuals** handling confidential medical, legal, or financial data

For these groups, accidental exposure of intent, interests, or vulnerabilities through AI-driven interfaces poses a real and growing threat.

Technical Outlook: Local AI and Real-Time Protection

The proposed system envisions a local privacy layer powered by lightweight LLMs such as Mistral 7B, which already run on most modern workstations (32–64 GB RAM). These models could enable:

- On-device detection of sensitive input and real-time overlay augmentation
- Dynamic obfuscation and decoy logic (Tree-of-Thought style)
- Intelligent result routing and masking within the orchestration layer

By avoiding cloud dependency, the system offers privacy-preserving workflows tailored for security-conscious users.

Although such augmentation is technically feasible, reliable implementation—especially DOM manipulation, prompt interception, and UI-level delays—is non-trivial. Many modern web apps use dynamic frontends (e.g., React), requiring resilient engineering to ensure consistency across sessions and platforms. For this reason, these capabilities are not part of the initial release, but are being explored for future development.

Systems with limited resources may fall back to simpler rule-based methods, such as keyword filters or DOM-based context tagging, which still offer value in less complex environments.

⚠ Always keep in mind

This system mitigates risks, 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.

🧩 For more control: API-Based Chatbot Mimicry

To overcome the limitations of DOM manipulation in dynamic web apps, the system can mimic chatbot interfaces using a controlled, API-connected frontend. This enables:

🔍 Prompt Filtering
Apply real-time sensitivity checks before sending prompts to the model.

🧪 Decoy Injection
Generate and route plausible decoy prompts to obfuscate user intent.

🖼 Overlay Augmentation
Add dynamic masking/demasking layers directly in the controlled UI.

🔀 Output Separation
Ensure only the real response is shown in the display slot; decoys remain background-only.

🖥 Built-In Delivery
Since the orchestration tool includes a locally hosted web interface, this functionality is available out of the box for users who need more control.

⚠ **Note:** The mimicked API interface may not replicate all features of the provider’s original web UI. Some elements like UI-specific memory, live suggestions, or threaded context may differ, depending on how the provider structures their API.

✅ 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).

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

🔧 The Future of Personal Infrastructure — Intelligent. Private. Local.

Our browser-based orchestration system unifies **digital workflows, smart environments, and AI-driven automation** — all locally hosted, privacy-respecting, and fully under the user’s control. No cloud dependency. No vendor lock-in.

While traditional automation relies on fixed routines — like *“doorbell rings → show camera feed”* — the real breakthrough lies in the system’s **built-in optimization layer** . It goes beyond basic cause-and-effect logic, enabling context-aware decisions that dynamically adjust both digital workspaces and connected smart home devices in real time.

To enable these physical interactions — such as adjusting lighting, blinds, or accessing LAN-only camera streams — users can integrate any browser-accessible **smart home platform** . A popular and well-suited option is **Home Assistant**, an **open-source** system that runs locally, offers a powerful browser interface, and communicates securely via local APIs. It fits seamlessly into the orchestration layer’s architecture, while maintaining full flexibility and user control.

🧠 Real Optimization in Action — With Smart Home Integration

🔀 Competing Priorities, Managed Intelligently

You’re in a deep work session. The doorbell rings, your child makes noise, and an AI-generated report begins processing.
→ The system allocates screen space to the camera feed, defers the baby monitor unless noise continues, and queues the report for silent review — without disrupting your flow.

🌗 Environmental Context Shaping

As you begin focused writing at dusk, the system dims lights, closes blinds via Home Assistant, and suppresses non-essential automations — keeping the mental environment aligned with the task.

📺 Scenes That Evolve With You

Start “Relax Mode”: lights soften, blinds close, music plays. Leave the room, and the system pauses playback, powers down smart lights, and waits until you return.
→ Scenes respond to presence — not just preprogrammed steps.

🛏 Smart Family Awareness

A child stirs late at night.
→ If you’re watching a movie, the LAN-only baby cam overlays discreetly. If you’re on a call, it appears silently on a secondary display.
→ One event, context-aware outcomes based on your current activity.

🔄 A Flexible, Controllable Optimization Layer

The optimization layer is not rigid or opaque. Users can:

- Toggle it off at any time
- Override it manually via voice or keyboard commands

- Replace it for specific scenarios where deterministic behavior is preferred

It is also intended to learn over time, storing preferences and command patterns locally — adapting its logic based on user intent and feedback, while keeping all data fully private.

For professionals managing business, home, and privacy-critical environments, this offers something truly new: A control layer that adapts like a real assistant — across screens, smart homes, and time.

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.

OpenGiraffe: A Comprehensive, Future-Proof Security Framework for Local AI Orchestration in High-Risk Environments

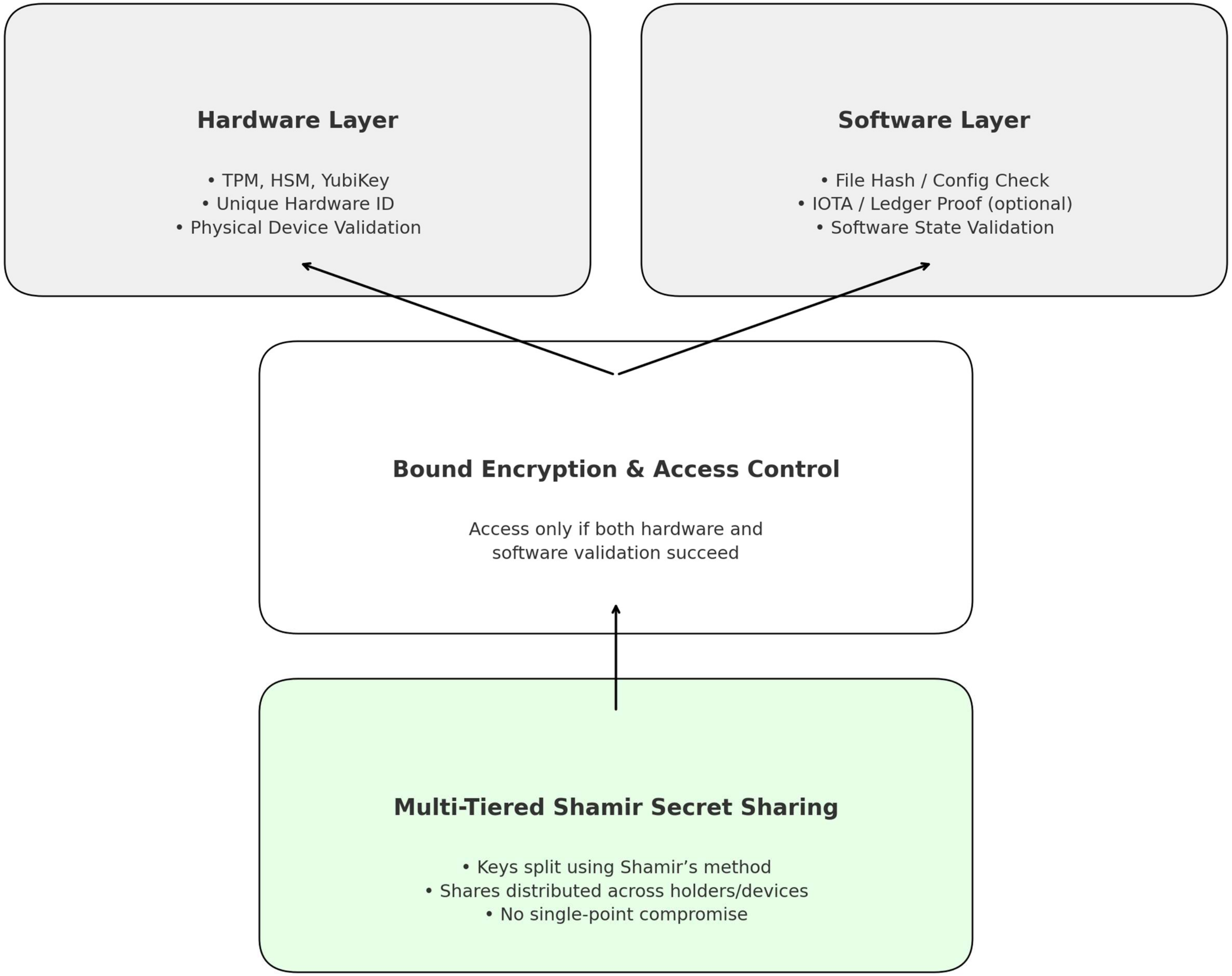
Abstract: **OpenGiraffe** is an open-source, modular security framework designed to enable fully local execution of AI-powered workflows while ensuring tamper-proof access control, immutable logging, and strict separation of data, roles, and configurations. The system is built specifically for high-risk environments where data integrity, long-term security, and operational resilience are paramount. The architecture is

designed to prevent not only current cyber threats but also future risks posed by emerging technologies such as quantum computing.

1. Purpose and Design Philosophy:
The rise of AI and Large Language Models (LLMs) in sensitive business processes introduces new vulnerabilities. **OpenGiraffe** exists to close these gaps by offering a framework where no unauthorized user can copy, decrypt, or misuse AI data—now or in the future. Even if encrypted containers are exfiltrated, their contents remain unusable due to:

- **By binding encryption to both hardware and software conditions—such as cryptographic file hashes or IOTA-based proofs—tamper resistance is enforced across physical and digital layers.**
- **Multi-factor authentication**
- **Ledger-based methods (blockchain, DLT) ensure integrity in connected systems, while air-gapped environments achieve similar guarantees through local cryptography and secure hardware (e.g., HSM, TPM).**
- **Cryptographic configuration freezing**
- **Revocable key material and user access**

Hardware- and Software-Bound Encryption with Shamir Secret Sharing



Hardware Layer

- TPM, HSM, YubiKey
- Unique Hardware ID
- Physical Device Validation

Software Layer

- File Hash / Config Check
- IOTA / Ledger Proof (optional)
- Software State Validation

Bound Encryption & Access Control

Access only if both hardware and software validation succeed

Multi-Tiered Shamir Secret Sharing

- Keys split using Shamir’s method
- Shares distributed across holders/devices
- No single-point compromise

2. Role-Based Data and AI Separation:
OpenGiraffe allows every employee to access the same locally hosted AI engine while strictly controlling:

- **The data each role can access (through separate vector databases in encrypted containers)**
- **The AI templates, prompts, workflows, and configurations presented to them**

This ensures that a researcher, HR staff, and finance controller each interact with the AI differently—both in what they can do and what information they can access. Department-specific containers keep sensitive information compartmentalized. The Locker App authenticates users, securely transfers role identifiers, and ensures only authorized configurations are loaded. User permissions can be revoked instantly, disabling all access in real time.

3. Advanced Security Layers (Optional):
OpenGiraffe supports multiple layers of defense, which can be combined to match an organization's risk profile:

- Device Fingerprint Binding: Ensures execution only on registered hardware.
- Multi-Factor Authentication (MFA): Adds OTP-based secondary authentication.
- Hidden and Departmental Containers: Strict separation of data, templates, and workflows for each department.
- Role-Based AI Customization: Ensures prompts, outputs, and automation are aligned to user roles.

- Tamper-Proof Configuration Freezing: Cryptographic hashes ensure that once a security configuration is deployed, it cannot be altered without detection. Any mismatch triggers immediate protective measures.
- Mandatory Logging with Auto Shutdown: In high-risk settings, every action must be logged—locally, to a management PC, or immutably to a ledger (e.g., IOTA). If logging is not possible, **OpenGiraffe** automatically locks down or shuts down to prevent any untraceable execution.
- Immutable Attempt Logging and Revocation: All events, including failed login attempts, role escalations, and configuration changes, are logged immutably. User access can be revoked instantly through ledger or network-triggered commands.
- Tiered Shamir Secret Sharing (SSS): For critical operations or emergency overrides, Shamir Secret Sharing ensures no single user can act alone. Multi-party authorization is enforced for accessing top-level containers, altering system settings, or releasing sensitive logs.

4. Hosting and Deployment Flexibility:
OpenGiraffe is hardware-agnostic and can be deployed:

- On local devices
- In on-premises secure server environments
- In private enterprise clouds
- In fully air-gapped deployments or in mixed environments with immutable rule sets

It is scalable from small teams to large enterprise environments with multiple departments.

5. Logging, Auditability, and Compliance:
The system enforces:

- Full traceability of every access, action, and change
- To ensure tamper-proof logging, secure storage can be implemented locally, across networks, or via blockchain/ledger solutions. On management devices, secure elements such as HSMs, TPMs or embedded chips can further protect logs from manipulation, even in isolated environments.
- Automated detection and enforcement of system lockdown if logging fails or is tampered with

All activities are cryptographically verifiable, providing strong compliance support for industries under strict regulation**.**

6. Optional Ledger Anchoring, Template Authenticity & Zero-Knowledge Proofs via IOTA

OpenGiraffe is not just built for today's security challenges. Its design anticipates future threats, including quantum decryption attacks. Even if encrypted containers are stolen, they cannot be decrypted without the original hardware, valid key material, and verified configuration. The OpenGiraffe project is committed to ongoing research into quantum-proof cryptography, Zero-Knowledge Proofs, and AI governance to ensure long-term security and innovation leadership.

OpenGiraffe optionally integrates **IOTA**—a scalable, feeless distributed ledger designed for verifiable, tamper-proof anchoring of events, code artifacts, and execution traces without exposing sensitive data.

By selectively anchoring cryptographic hashes of:

- WR Code payloads
- Geofence presence proofs
- AI agent decisions
- Orchestration templates and automation scripts
- Workflow execution snapshots
- User activity events (where compliance requires)

OpenGiraffe enables: ☑ Verifiable claims without centralized trust
☑ Tamper-proof logging of AI-driven actions and template integrity
☑ Immutable timestamping of sensitive events
☑ Optional privacy-preserving user activity audits

Why IOTA?

- **Feeless Transactions:** Scalable even for high-frequency micro-events.
- **Tangle Architecture:** Energy-efficient, lightweight, and parallelizable.
- **Rich Metadata Support:** Ideal for anchoring multi-agent decision chains and human-in-the-loop events.

- **Future-Ready for Post-Quantum Security:** Supports hash-based quantum-resistant signatures without protocol disruptions.

This broader use of IOTA within OpenGiraffe extends beyond just WR Code authenticity. It provides the foundation for:

- Verifiable **template authenticity**: Ensuring that orchestrated actions stem from trusted, unchanged templates.
- Confirmable **code originality**: Proving that automation logic has not been altered or manipulated.
- Transparent **logging of AI decisions and outputs**: Allowing both machine and human participants to audit past actions.
- Optional **user activity logging**: With strict privacy controls, providing tamper-proof documentation of who initiated which actions in sensitive environments.

To encourage adoption, OpenGiraffe promotes the idea that WR Code providers and enterprises can voluntarily anchor their hashes on IOTA Layer 1, giving end-users and regulators the ability to independently verify authenticity, originality, and integrity. This could help create an ecosystem of higher trust without relying on centralized control structures.

However, while a cryptographic hash secures the integrity of any payload, true identity verification of the publisher still requires anchoring via identifiable wallets or decentralized identities (DIDs). This dual approach—content integrity plus source authenticity—allows for full trust without sacrificing privacy.

Unlike traditional blockchains such as **Bitcoin, Ethereum, or Solana**, IOTA's architecture is inherently more adaptable for post-quantum upgrades. Its feeless nature also makes it practical for the high volume of micro-interactions typical in AI orchestration.

Integration Scenarios:

1. **QRGiraffe:** Secure WR Code processing with optional IOTA logging for auditability.
2. **Geofence Proofs:** Anonymous presence proofs with optional ledger anchoring.
3. **AI Decision Trails:** Verifiable record of orchestrated AI-driven actions, including template verification.
4. **Optional Future Incentives:** Crypto rewards, NFTs, or smart contract automation where applicable.
5. **User Consent & Activity Logging:** Tamper-proof logs of user-triggered actions for compliance-heavy sectors.

All IOTA features are strictly **opt-in**, fully **local-first**, and **privacy-preserving**. Sensitive data is never stored—only cryptographic proofs are optionally shared for independent verification.

7. Optimando Service Offering:
While OpenGiraffe is freely deployable as open source, Optimando provides:

- **Expert consulting on AI orchestration security**
- **Customized role and container design**

- **Deployment of secure, tamper-proof environments**
- **Hands-on staff training and compliance support**

8. Conclusion:
OpenGiraffe provides a holistic, future-proof solution for secure AI deployment. With role-based separation, immutable configuration, hardware-tied encryption, mandatory logging, and system lockdowns, it ensures that AI can be safely used in the most sensitive environments. Through optional advanced security features and **Optimando's** consulting support, organizations can adopt AI with confidence—without compromising on privacy, operational integrity, or long-term data protection.

OpenGiraffe and its tools are open-source and trustless—enabling everything from easy setups to tamper-proof enterprise solutions. Future-ready LLMs without blind trust.

LetMeGiraffeThatForYou: Visual Question Capture and Automated AI Orchestration Trigger in OpenGiraffe

As part of the **OpenGiraffe** open-source AI orchestration framework, we introduce "**LetMeGiraffeThatForYou**"—a decentralized feature that allows users to instantly capture any part of their digital environment and trigger an AI-powered research process with zero manual input, simply by clicking a button.

How It Works (with Privacy-First Manual Control):

1. A floating capture button is available inside the browser, workspace, or any **OpenGiraffe**-integrated environment.

2. When clicked:
    - The user can manually select an area of the screen or capture the full window—for example, a chat question, social media post, or any on-screen content.
3. The captured input—whether screenshot, screencast, or structured data—is passed into **OpenGiraffe's** master orchestrator, where:
    - AI automatically detects the question or intent from the selected content.
    - A predefined AI research workflow is triggered, running through local models, cloud services, or both, depending on user preferences.
4. The system then generates:
    - A context-aware answer or insight.
    - A branded, share-ready PDF, visual snapshot or a video capture, automatically copied to the clipboard for instant pasting into any chat, post, or document.

🛡 Optional Manual Trigger for Maximum Privacy:

- **OpenGiraffe** is designed with a continuous optimization layer that can automatically monitor digital activities (e.g., tab context, page content) to suggest proactive AI assistance when enabled.

- For users or organizations who prefer strict control or offline operation, this continuous layer can be toggled off.

- Users explicitly choose when and what to capture (screenshot, screencast, text).
- No background monitoring or context capture takes place unless initiated by the user.

The process is triggered only when explicitly requested for selected input types, ensuring that users retain full control over when and how AI-driven analyses are initiated. This is particularly relevant for users who choose to toggle off the automatic optimization layer within **OpenGiraffe**: in such cases, **LetMeGiraffeThatForYou** allows users to manually capture input and selectively trigger AI research only when needed. In team discussions, online chats, or collaborative workspaces where questions naturally arise, **LetMeGiraffeThatForYou** enables users to provide instant, AI-assisted answers through a simple click-and-share mechanism—making knowledge sharing effortless, efficient, and privacy-friendly.

Key Features of LetMeGiraffeThatForYou

• 🖱️ **User-Defined Capture Area:**
Users can select the **exact portion of their screen** to capture—whether it’s a single question, an entire chat window, or a complex discussion thread. Flexible for **full screens, snippets, or specific visual content**.

• 🧠 **Automatic Intent Recognition:**
The system’s AI **autonomously detects the last question, request, or informational need** from the captured content—without requiring manual prompting (depending on the selected workflow template).

- 🔄 **Preconfigured AI Research Workflows:**

Upon capture, the system triggers a **predefined OpenGiraffe orchestration**, executing **parallel AI research agents** to synthesize accurate, context-aware answers. Research can run **fully locally** or leverage **hybrid cloud models**, as configured by the user.

- 🛡️ **Privacy-First & Decentralized by Design:**

All processing can operate **100% locally** with **no centralized data storage or tracking**. Alternatively, users can **opt-in to cloud AI** for enhanced capabilities. Full control remains with the user or organization at all times.

- 🚀 **Instant Visuals for Viral Sharing:**

LetMeGiraffeThatForYou creates **automatically branded, shareable visual outputs** (PDFs, screenshots, video snippets, or even audio messages) that are copied automatically into the clipboard and can be **pasted directly into chats, posts, or documents**—enabling seamless knowledge-sharing.

Strategic Position: The domain **LetMeGiraffeThatForYou.com** has been secured to protect the naming rights of this concept, which refers to a built-in feature within the **OpenGiraffe** orchestration framework. The underlying concept, technical design, and process constitute original intellectual property (IP) within the **OpenGiraffe** ecosystem. By combining customizable visual capture, automated question detection, and orchestrated AI research, **LetMeGiraffeThatForYou** introduces a playful yet powerful way to simplify digital interactions, enhance knowledge sharing, and promote AI adoption—without reliance on centralized infrastructure, if desired.

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

⚖️ Strategic Trade-Offs and Professional Target Group

Optimando AI's OpenGiraffe architecture is designed around **autonomy, data privacy, and advanced user-controlled automation**. As a concept, it acknowledges several strategic trade-offs that professional users and potential investors should be aware of:

- **Higher Token and Context Costs**: Workflows involving large prompts, multi-agent logic, or external reasoning (e.g., cloud LLMs) can lead to increased token usage. However, these costs are steadily decreasing as open models improve.
- **Local Resource Requirements**: Running local LLMs and orchestration logic requires capable hardware (e.g., sufficient RAM and CPU/GPU). While often more efficient than cloud alternatives for burst workloads, continuous usage still implies measurable local energy use.
- **Advanced Setup Needs**: The system assumes a professional or technically literate user base. Users manage their own storage, security (e.g., encryption), and model selection.

These are not flaws, but deliberate design trade-offs in favor of **data control, workflow transparency, and security**. Most cloud-first systems offer convenience but at the cost of vendor lock-in and profiling risks. Optimando.ai is conceptually positioned for **entrepreneurs**, **researchers**, **educators**, and **technically skilled professionals** — users who gain long-term strategic value from maintaining control over their data and automation stack. For these users, the **advantages clearly outweigh the operational complexity**.

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 ***OpenGiraffe*** – 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 Marinertechcrunch.comtechcrunch.com, OpenAI’s Operatortechcrunch.comtechcrunch.com, UCSD’s Orca browser researcharxiv.orgarxiv.org, the Nanobrowser open-source projectgithub.comgithub.com, Opera’s AI initiatives (Aria and Neon)press.opera.compress.opera.com, and curated lists of web AI agents and automation toolsgithub.comgithub.com. Each of these informs the feature comparison and underscores the novelty of optimando.ai’s approach.

☑ Novel Innovations of *optimando.ai*

⚙ Real-Time Contextual Optimization Layer

An **open-source, browser-based orchestration tool** that continuously interprets **user behavior, screen context, and intent** to dynamically route tasks to specialized AI agents — enabling **real-time assistance without requiring explicit commands**.
optimando.ai proactively supports the user based on their current activity — such as drafting, summarizing, or researching — rather than relying solely on typed input.
Users can initiate multi-step automations via natural language, speech commands, or automatically detected user intent — including instructions to display results in designated visual slots within the interface. Display

slots can also be allocated dynamically based on contextual relevance and priority, allowing high-urgency or decision-critical outputs to automatically surface in more prominent or persistent positions.

The system responds dynamically by monitoring the workspace (e.g. tab content, interaction patterns) and coordinating agents in real time.

This represents a **novel open-source approach to AI-driven workflow orchestration**, combining live context awareness, voice-driven control, and visual task management in a unified, browser-centered environment.

. 🛡 Integrated Privacy Layer with Risk Mitigation

On-device logic detects sensitive content, delays or masks prompt execution, and can optionally inject decoys to obscure real user intent.

This layered approach to privacy — combining detection, execution control, and obfuscation — is not present in mainstream tools. It addresses profiling risks directly at the orchestration level, which most cloud- or agent-based systems do not offer.

🔐 Air-Gapped, Multi-Agent Orchestration (No Cloud Dependency)

A fully local deployment option where multiple AI agents run in coordination without internet access, suitable for high-security or regulated environments.

While platforms like **LangChain**, **AutoGPT**, **Cognosys**, **Lindy.ai**, **Superagent**, **AgentOps**, **Microsoft AutoGen**, and various open-source agent frameworks offer components such as agent coordination, multi-step automation, or tool integration, they typically lack one or more of the truly distinctive features that *optimando.ai* provides.

In particular, while some tools offer local model integration or agent chaining, *optimando.ai* uniquely provides an **open-source, plug-and-play multi-agent orchestration system** that can run **fully offline if needed** — including **real-time context interpretation**, **contextual intent support**, **speech-triggered automation**, and a **visual slot-based interface** for task execution.

This combination of capabilities is **currently unmet** in other AI automation offerings, positioning *optimando.ai* as a **novel solution for privacy-sensitive, highly interactive, and locally controlled AI-driven workflows**.

These three features — especially in **combination** — position *optimando.ai* as a **privacy-focused, locally-executable orchestration system with intelligent agent coordination**, which is **currently unmatched** in the browser-based LLM tooling space.

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.**

Tree of Thought
Analytics
46%
87.8%
5%
15%
9.9%
15%
6%
AI Orchestration
Agants
Task Flow
Sane 7
Tmsa 6
Tasck 9
Taus 6

🧠 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.

💡 Responsive Output via Smart Buffering

To ensure a fluid user experience, the output coordinator buffers AI results before displaying them in visual slots. Only complete, relevant outputs are shown, keeping the interface responsive and distraction-free. As

AI performance and computation capabilities improves over time, this buffer time will shrink — moving the experience closer to true real-time interaction, even in complex multi-agent scenarios.

🛠 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.

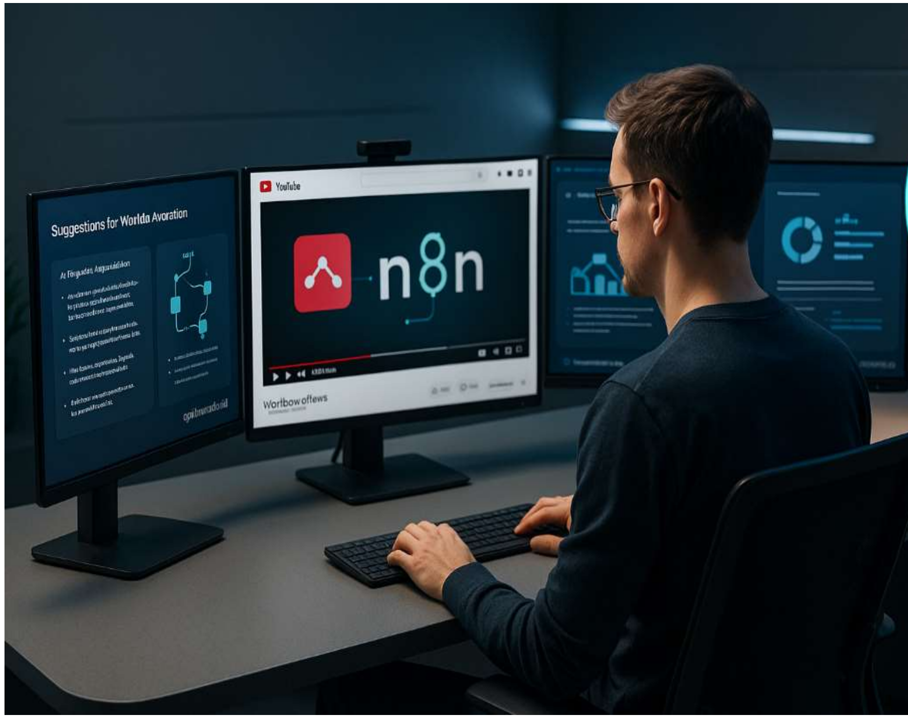
This deployment strategy is particularly useful in environments with strict data governance policies, limited or no internet access, or where offline, local AI processing is required.

🏁 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.org 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 ENABILING 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.

This work documents the OpenGiraffe© architecture, including QRGiraffe© — an open-source framework for browser-based AI orchestration developed by Optimando.ai.

OPTIMANDO AI