Agentical.Net: Private AI Computing Space

by Team—Code 202

www.agentical.net

Abstract

The rapid adoption of generative AI has been driven by centralized cloud infrastructure, where model inference is executed on provider-controlled systems and user data is externalized by default. While this approach enables scalability and convenience, it introduces structural risks related to privacy, governance, regulatory compliance, and long-term dependency. As AI systems become embedded in sensitive, regulated, and mission-critical workflows, these risks increasingly outweigh the benefits of centralized execution. To solve this problem *Agentical.Net* is offers, a decentralized, local-first AI platform that reframes inference as a user-controlled system capability. It combines local GPU-based inference, encrypted peer-to-peer communication, a local knowledge (RAG) layer, and an agent coordination layer based on the Model Context Protocol (MCP). Together, these components enable private, inspectable, and sovereign AI workflows in which prompts, data, and execution remain under user control by design. By treating inference locality, explicit coordination, and decentralized communication as foundational principles, *Agentical.Net* is moving towards practical framework for building AI systems that are resilient, compliant, and aligned with the realities of modern computing environments.

Contents

1 Introduction

The rapid adoption of generative AI has been driven largely by centralized cloud infrastructure. Large-scale data centers now act as the primary locus for model hosting, inference execution, and data processing. While this architecture has enabled rapid deployment and scalability, it has also created a structural imbalance: users gain access to powerful models at the cost of relinquishing control over data, execution context, and system behavior.

This imbalance is no longer a secondary concern. As AI systems are increasingly embedded in professional workflows, regulated environments, and personal knowledge processes, questions of data exposure, jurisdictional control, auditability, and dependency risk become fundamental architectural issues rather than policy afterthoughts. Inference—the moment where user data is actively processed by a model—represents the most sensitive and least transparent point in this pipeline.

Agentical.Net addresses this challenge by rethinking where and how AI inference occurs. Instead of treating inference as a remote service, Agentical treats it as a local operation executed on hardware the user already owns or directly controls. Encrypted peer-to-peer networking replaces centralized routing, and identity and permissions are enforced at the system level rather than through institutional trust in third-party providers.

1.1 The Problem With Modern AI Infrastructure

Modern AI infrastructure is predominantly built around centralized cloud execution. In this model, model weights, inference pipelines, orchestration logic, and often user data are consolidated within hyperscale data centers operated by a small number of providers. While this architecture simplifies deployment and enables elastic scaling, it also embeds a set of structural assumptions that increasingly conflict with the realities of how AI systems are used.

At the core of the problem is control asymmetry. Users interact with AI systems through APIs or managed interfaces, but the execution environment—where inference actually occurs—remains opaque and externally governed. Prompts, intermediate representations, and outputs are transmitted to remote systems where they may be logged, retained, inspected, or repurposed, often in ways that are difficult to audit or meaningfully constrain. *Even when providers offer contractual assurances, technical enforcement remains outside the user's domain of control.*

This architecture matters because inference is the point of maximum data sensitivity. Unlike model training, which may involve aggregated or anonymized datasets, inference operates directly on live user inputs: personal data, proprietary documents, internal communications, and context-specific knowledge. Centralizing inference therefore centralizes risk. Breaches, misconfigurations, insider access, or jurisdictional mandates can all expose data in ways that are structurally unavoidable within cloud-based execution models [5, 4, 2].

A second limitation is jurisdictional and regulatory fragility. Centralized AI platforms are subject to the legal regimes of the regions in which they operate. For organizations operating across borders — or within highly regulated sectors — this creates uncertainty around data residency, lawful access, and compliance with frameworks such as GDPR. Attempting to address these issues through policy layers, regional deployments, or contractual controls adds complexity but does not change the underlying execution model: inference still occurs on infrastructure the user does not own [3].

Modern AI infrastructure also introduces systemic dependency and lock-in. As applications become tightly coupled to specific cloud providers' models, APIs, and pricing structures, switching costs increase and architectural flexibility decreases. Performance, availability, and cost predictability are all externalized to third parties. For individual users and smaller organizations, this dependency can be as consequential as it is invisible, shaping what kinds of AI workflows are economi-

cally or technically viable.

Finally, centralized AI infrastructure assumes that compute must be remote to be useful. This assumption was historically justified by the scarcity and cost of capable hardware. However, it no longer reflects current conditions. Consumer and professional devices increasingly ship with GPUs capable of meaningful inference workloads. Browsers now expose standardized GPU interfaces, and peer-to-peer networking technologies have matured to support secure, low-latency communication without centralized routing. *The persistence of centralized inference is therefore less a technical necessity than an inherited architectural convention.*

In aggregate, these factors reveal a mismatch between how AI infrastructure is currently designed and how AI systems are increasingly expected to behave: **private by default, auditable by design, resilient to external control, and adaptable to diverse execution environments.** Addressing this mismatch requires more than incremental improvements to cloud platforms. It requires re-examining the foundational assumption that AI inference must occur somewhere other than where the user is.

For many industries, the limitations of current AI infrastructure are decisive. Legal, medical, financial, high-tech, research, and government organizations operate under strict requirements for data handling, confidentiality, and auditability. In these environments, sending sensitive information to external servers may violate regulations or internal policies. As a result, AI adoption is often delayed, restricted, or blocked entirely, despite clear potential benefits. This creates a paradox. The sectors that could gain the most from advanced AI are often the least able to use it safely.

As AI expands into healthcare, law, government, and enterprise, risks and constraints become unacceptable. Many organizations now seek alternatives that allow them to retain full ownership and control of the data they process. With unpredictable billing, hidden fees for bandwidth, inference, or storage, and amid rising enterprise GPU costs, a growing ecosystem of developers, creators, and small businesses are turning to consumer-grade GPUs — such as the NVIDIA RTX 3060, 3080, 3090, and 4090 — as viable alternatives for AI workloads.

1.2 The Way Forward

Addressing the limitations of modern AI infrastructure does not require incremental policy adjustments or stronger contractual assurances layered on top of centralized systems. It requires shift in where inference occurs, how data moves, and who controls the execution environment. The way forward lies in treating AI inference as a local, user-controlled operation, supported by decentralized communication rather than centralized orchestration.

This shift begins by re-centering **inference on hardware that users already own or directly control**. Advances in consumer and professional GPUs, combined with standardized local compute interfaces, make it increasingly practical to execute meaningful inference workloads outside of data centers. When inference runs locally, sensitive inputs never need to leave the user's environment by default, fundamentally reducing exposure risk and eliminating entire classes of data governance concerns rather than attempting to manage them after the fact.

Equally important is the replacement of centralized routing with secure peer-to-peer communication. Rather than funneling all requests through provider-controlled infrastructure, modern networking protocols allow devices to establish encrypted, direct connections with one another. This model preserves low latency and scalability while removing the requirement that intermediaries observe, store, or mediate inference traffic. Trust boundaries become explicit and technically enforced, rather than implicit and contractual.

The way forward also requires rethinking identity, permissions, and coordination in decentralized systems. Without a central author-

ity managing execution, systems must embed mechanisms for authentication, access control, and agent-level governance directly into the architecture. These mechanisms must operate consistently across devices, networks, and usage contexts, enabling collaboration without reintroducing centralized control points.

Crucially, this approach does not reject cloud infrastructure outright. Instead, it repositions the cloud as an optional component rather than a mandatory execution layer. Models, tools, and agents can be shared, monetized, synchronized, or discovered through decentralized means, while inference itself remains local and sovereign. This inversion of the traditional AI stack — local execution first, remote services second — aligns infrastructure design with emerging expectations around privacy, resilience, and autonomy.

Agentical.Net is built around this architectural reorientation. Rather than optimizing centralized inference pipelines, it is designed to enable local AI execution connected through encrypted peer-to-peer networks, with system-level support for identity, permissions, and collaboration.

With its privacy-preserving platform *Agentical.Net* enables users to run and interact with advanced generative AI models directly through peer-to-peer GPU computing, allowing individuals and businesses to leverage their own hardware (or tap into a global pool of shared, idle GPUs) to power AI tasks securely and efficiently.

Agentical.Net **is designed to break down technical complexity, eliminate unnecessary intermediaries, and empower users with control over their data, their compute, and their AI experience. The platform marks a shift toward a more transparent, equitable, and user-centric AI infrastructure — one where privacy, affordability, and accessibility are foundational principles**.

2 The Agentical.Net

2.1 What is Agentical.Net

At its core *Agentical.Net* represent decentralized AI execution platform designed to enable **private, user-controlled generative AI workflows**, providing the infrastructure necessary to run AI inference locally — *on hardware owned or directly controlled by the user* — while still supporting collaboration, sharing, and coordination through secure peer-to-peer communication.

Unlike conventional AI platforms that treat inference as a remote service accessed through centralized APIs, *Agentical.Net* treats inference as a local system capability. Models execute within the user's environment, data remains resident on the user's device by default, and network communication occurs only when explicitly required. This design directly addresses the structural issues identified in modern AI infrastructure: loss of control over execution context, unnecessary data exposure, and dependency on centralized intermediaries.

The platform is built on open web and networking standards, allowing it to operate across devices and environments without proprietary runtime dependencies. Local computation is enabled through standardized GPU interfaces, while encrypted peer-to-peer channels facilitate direct communication between agents, users, or collaborating systems. This combination allows *Agentical.Net* to function as a distributed AI substrate rather than a hosted application.

From a systems perspective, *Agentical.Net* occupies a distinct position in the AI stack. It is not a model provider, a cloud hosting service, or a development framework. Instead, it acts as an execution and coordination layer that sits between AI models and end users, governing how inference is performed, how data

moves, and how trust is established across participants. Models can be local, shared, or externally sourced; what remains consistent is that inference control stays with the user.

Agentical.Net is also **explicitly designed to be usable without programming expertise**. By abstracting protocol management, device coordination, and execution control into a user-facing environment, the platform enables individuals, teams, and organizations to use private AI workflows without constructing custom infrastructure. This usability constraint is not incidental — it is a deliberate design choice intended to ensure that privacy-preserving AI execution is accessible beyond highly specialized technical teams.

With its **privacy-first, peer-to-peer** AI platform *Agentical.Net* falls into a new category of AI infrastructure: **sovereign AI**. *Agentical.Net* approaches AI as a private mesh of local capabilities, meaning no more renting intelligence from a cloud provider — just operate and delegate it yourself.

2.2 Architecture Overview & Core Technologies

At a conceptual level, *Agentical.Net* is built as a peer-to-peer AI platform. We've built a platform around a number of core technologies and tools that work together to deliver private, scalable, and user-controlled AI. *Agentical.Net* combines private execution, encrypted peer-to-peer communication, and autonomous agents into a cohesive workflow that supports both simple interactions and complex, long-running AI processes.

The architecture is composed of five primary layers:

- A local execution layer responsible for model inference and agent runtime.
- A peer-to-peer networking layer enabling secure device-to-device communication.
- A local knowledge layer for private data access and retrieval *or* **RAG**.
- An agent coordinating workflows, tools, and memory layer *or* **MCP**.
- A coordination layer for sharing and monetization.

Crucially, none of these layers depend on centralized inference or data processing. Central services, where present, are limited to coordination and never handle model inputs, outputs, or intermediate data. Our architecture is intentionally scoped to **inference-time execution and coordination**. It does not attempt to replace model training pipelines, hardware drivers, or operating system security primitives, but it focuses on the boundary where AI systems interact with users and data thus ensuring that this boundary is local, explicit, and technically enforceable. That means that in this model, the browser or local runtime becomes the execution environment, the GPU becomes a shared but user-controlled resource, and the network becomes an optional connector rather than a mandatory intermediary.

2.2.1 Local execution layer

The effectiveness of a local-first AI execution model depends critically on how computation is exposed to user-space applications. Historically, high-performance GPU compute has been gated behind platform-specific APIs, privileged drivers, or native runtimes, limiting portability and making secure sandboxed execution difficult. For *Agentical's* local compute layer to be viable across devices and environments, it requires a standardized, safe, and performant interface to modern GPUs.

WebGPU fulfills this role. It provides a low-level, cross-platform GPU API designed explicitly for general-purpose compute and machine learning workloads, exposed in user-space environments such as the browser. In *Agentical's* architecture, WebGPU serves as the primary execution substrate that enables local AI inference without requiring proprietary runtimes or elevated system privileges.

WebGPU is a cutting-edge web standard developed by the W3C to provide modern, low-

level access to GPU hardware directly from web browsers[6]. Unlike its predecessor, WebGL, WebGPU is designed for both advanced graphics and general-purpose GPU computations (GPGPU). This dual capability enables developers to execute complex tasks such as machine learning inference, and AI computations directly within the browser environment. By bringing native GPU acceleration to the web, WebGPU makes real-time AI more accessible to developers and users alike, marking a major step toward a truly AI-ready web and with its approach offers key some advantages[7], [1].

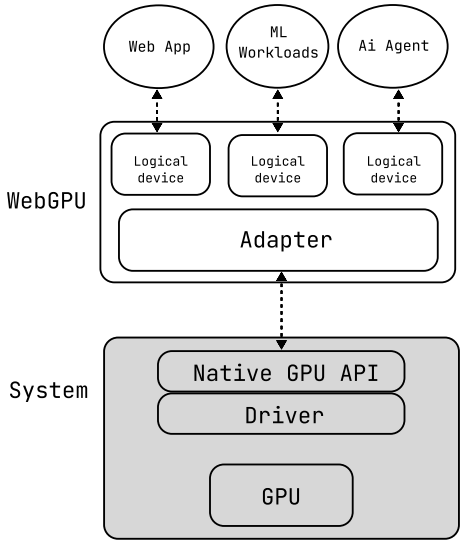

Figure 1: Structure of a web application using WebGPU

This matters for three reasons:

1. **Inference can run where the user is** — WebGPU allows AI workloads to execute directly on the user's device, collapsing the execution boundary to the local system. This aligns with *Agentical's* goal of eliminating unnecessary data movement during inference.

2. **Compute access is standardized and sandboxed** — WebGPU is defined by an open specification and implemented by browser vendors on top of native GPU backends. Applications gain access to GPU acceleration without raw device control, reducing attack surface while preserving performance.

3. **The browser becomes a viable AI runtime** — By making GPU compute a first-class web capability, WebGPU transforms the browser from a thin client into a legitimate local execution environment. This enables *Agentical.Net* to deliver private AI workflows without native installations or platform-specific binaries.

Within *Agentical.Net*, WebGPU is treated as a strategic enabler and it allows the platform to:

- Execute inference locally across heterogeneous devices.
- Maintain strict data locality guarantees.
- Avoid dependence on proprietary GPU runtimes.
- Deliver AI capabilities through a browser-accessible environment.

By anchoring the local compute layer on WebGPU, *Agentical.Net* ensures that private,

sovereign AI inference is not confined to specialized setups but can operate wherever modern computing environments exist.

2.2.2 Peer-to-peer compute layer

While local compute ensures that AI inference remains under user control, it does not by itself enable collaboration, coordination, or shared workflows. The role of the peer-to-peer communication layer is to connect locally executing processes and users directly, without routing data through centralized servers or exposing inference traffic to third-party intermediaries.

In *Agentical.Net*, this layer is implemented using **WebRTC**, which provides the transport, encryption, and connectivity primitives required for secure, low-latency, device-to-device communication. This layer is essential to *Agentical.Net* architecture because privacy and sovereignty are not preserved if local execution is paired with centralized networking. Data locality must be enforced end to end — both at execution time and during communication.

WebRTC is a mature, widely deployed networking technology designed for real-time communication across diverse network conditions. Unlike many networking protocols where encryption is optional or layered on top, WebRTC cannot operate without encryption. Every WebRTC connection in *Agentical.Net* uses two mandatory security mechanisms: DTLS *(Datagram Transport Layer Security)* for key exchange and authentication and SRTP *(Secure Real-time Transport Protocol)* for encrypting data payloads. These mechanisms apply regardless of whether the traffic is sent directly between peers or relayed through a TURN server. Once a WebRTC connection is established, all data transmitted between peers is encrypted on both ends, and encryption keys are negotiated directly between the peers using DTLS. No intermediary, including signaling or relay servers, has access to these keys.

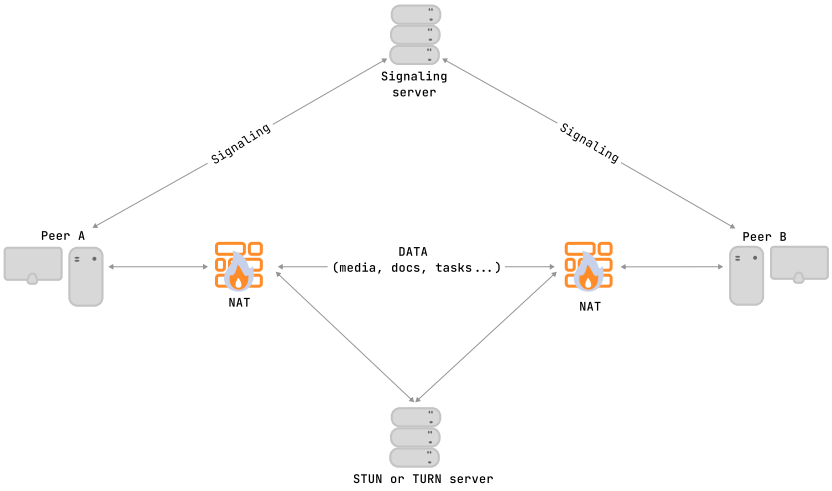

Figure 2: Secure P2P connection via WebRTC

Agentical.Net uses WebRTC data channels to support a constrained but intentional set of interaction patterns:

- One-to-one agent communication.
- Small-group collaboration among explicitly authorized peers.
- Session-based exchanges tied to specific inference or coordination tasks.

Connections are ephemeral by design. Once a session ends, no persistent communication

state is retained by the platform. This minimizes long-lived exposure and simplifies both security analysis and compliance considerations. The peer-to-peer layer is explicitly **not** a message queue, storage system, or synchronization backend. Persistent data remains local unless users intentionally export or replicate it, which means that:

- Prompts, model inputs, embeddings, and outputs are never visible to third parties.
- Relay servers cannot inspect or modify payloads.
- Man-in-the-middle attacks are prevented by cryptographic verification.

2.2.3 A local knowledge layer

The purpose of the local knowledge layer — commonly referred to as a **Retrieval-Augmented Generation (RAG) layer** — is to provide AI systems with access to this private data **without externalizing it**. In *Agentical's* architecture, the RAG layer operates entirely within the user-controlled environment, ensuring that data retrieval, indexing, and contextual injection occur locally and remain subject to the same sovereignty guarantees as inference itself.

In many cloud-based AI platforms, RAG is implemented as a centralized service. Documents are uploaded to provider-managed storage, indexed remotely, and queried during inference. While this approach is convenient, it reintroduces the same structural risks that local inference is designed to eliminate: data exposure, opaque retention policies, and dependency on external infrastructure.

A local RAG layer matters because knowledge is often more sensitive than prompts. Private documents may include intellectual property, regulated information, or personal records that cannot be safely externalized. Even when encrypted at rest or protected by policy, centralized storage creates aggregation points that are inherently attractive targets and subject to jurisdictional control.

By contrast, a local RAG layer ensures that:

- Private data never leaves the user's environment by default.
- Retrieval occurs under explicit user control.
- Contextual augmentation is transient and task-scoped.
- Knowledge access can be audited and constrained technically.

Within *Agentical.Net*, the local knowledge (RAG) layer acts as a **bridge between static private data and dynamic AI reasoning**. It enables powerful, context-aware AI workflows without requiring that sensitive knowledge be externalized or permanently embedded into models. Combined with local inference and peer-to-peer communication, the RAG layer completes a core architectural triad:

- **Local compute** ensures execution sovereignty.
- **Peer-to-peer networking** enables secure coordination.
- **Local knowledge retrieval** enables private, contextual intelligence.

Together, these layers allow *Agentical.Net* to support sophisticated AI use cases (research, analysis, collaboration, and automation) while preserving the foundational guarantees of privacy, control, and architectural integrity.

2.2.4 Agent coordinating layer

As AI systems shift from single-turn exchanges to continuous, goal-oriented workflows, coordination moves to the forefront. Models on their own do not handle extended tasks, track and reason over evolving state, or determine when to pull in external knowledge, call tools, or interact with other agents. These duties fall to an **agent layer** — a control structure that manages inference, context, and actions over time.

This role is fulfilled by the **MCP (Model**

Context Protocol) layer. The MCP layer provides a structured, explicit mechanism for coordinating workflows, tools, memory, and knowledge access across locally executing agents, without embedding this logic directly into models or centralizing control in external services.

Without an agent layer, AI systems remain reactive and brittle. Each interaction must be manually framed, context must be reconstructed repeatedly, and integration with external tools or data sources becomes ad hoc. More critically, embedding coordination logic directly into models or prompts obscures control flow and weakens security boundaries. These issues are addressed by the MCP layer by:

- Separating **reasoning** (model inference) from **coordination** (agent control).
- Making context, tools, and memory **explicit system components**.
- Enabling **multi-step, goal-oriented workflows** that persist across interactions.
- Preserving privacy and locality by keeping coordination close to execution.

This matters because in privacy-preserving, decentralized systems, orchestration cannot be outsourced to opaque backends. It must be **locally enforceable and inspectable**.

Within *Agentical.Net*, MCP functions as the **agent coordination layer** that sits above local compute, peer-to-peer communication, and the local knowledge (RAG) layer. It does not execute inference itself, nor does it own data. Instead, it governs *when* and *how* other components are invoked. That means that MCP is responsible for important tasks, such as:

- Selecting and invoking tools or knowledge sources.
- Assembling structured context for inference.
- Coordinating interactions with other agents or peers.

2.2.5 Sharing and monetization layer

In settings where multiple agents run in parallel (either on a single device or over peer-to-peer links), the MCP layer offers a unified framework for coordination. Each agent preserves its own workflow and memory context, even when it shares inference outputs, hands off subtasks, or engages in collaborative sessions. The MCP layer guarantees that agent boundaries stay clearly separated, avoiding unintended state exposure or unauthorized privilege escalation between agents.

The goal of agent monetization and sharing in *Agentical.Net* is to facilitate the **controlled distribution of agent capabilities**. Our platform fully supports both sharing and monetizing agents while maintaining its core guarantees: local execution, explicit permissions, and user sovereignty.

Most existing AI "marketplaces" are built around centralized execution. Users purchase access to hosted models, prompts, or tools that run on provider infrastructure. While convenient, this model conflates *ownership* with *access* and requires users to trust that execution, data handling, and usage boundaries are enforced correctly.

Agentical.Net approaches monetization differently. Because agents execute locally and are coordinated through MCP, they can be shared as **portable coordination logic**, not as remotely hosted services. This enables us to support multiple sharing modes, each with distinct trust and control properties, like:

1. **Private Sharing:** Agents can be shared directly between trusted peers using peer-to-peer communication — no central registry is required, and shared agents remain under the recipient's control once received.
2. **Collaborative Sharing:** In collaborative contexts, multiple users may co-develop or co-maintain an agent, where MCP enables clear ownership and separation between agent logic and local

data. This supports iterative improvement without exposing private environments.

3. **Public Distribution and Monetization:** For broader reuse, agents may be published to a decentralized marketplace or registry. Crucially, publishing an agent does not grant the creator visibility into how or where it is used, unless the user explicitly opts in.

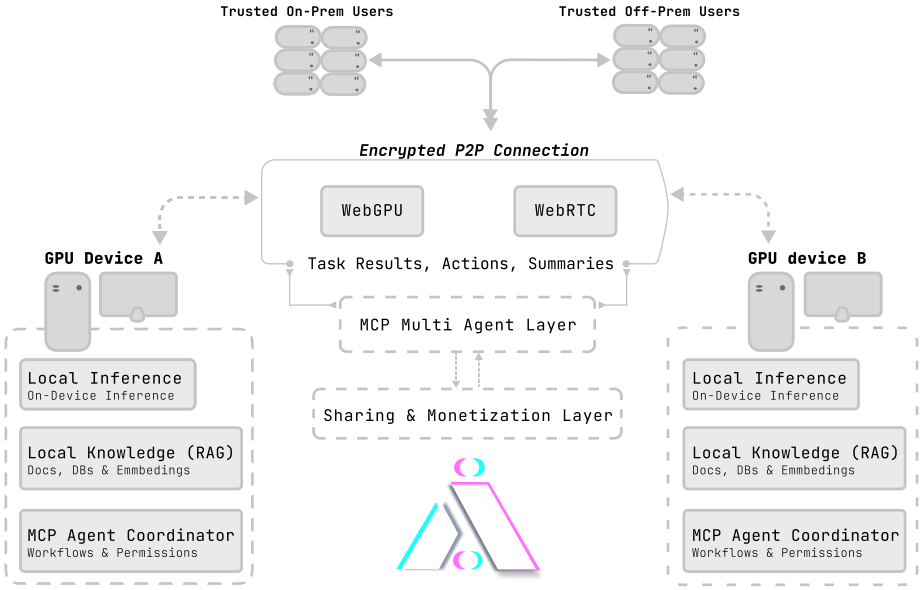

Figure 3: Architecture of Agentical.Net Workflow

2.3 AI Inference

AI inference is the operational core of any artificial intelligence system. It is the moment when a trained model processes input data and produces an output, like answering a question, generating text, analyzing a document, or driving an automated action. While much attention is given to model training, **inference is where AI actually interacts with the real world**, and where most of the cost, risk, and architectural consequences occur. Inference operates on live, contextual user data — often personal, proprietary, or regulated—and directly influences downstream actions and decisions. As such, the inference pathway defines the effective privacy boundary of the system.

In centralized architectures, inference is inseparable from external infrastructure. Inputs must be transmitted to third-party environments, executed under provider-defined conditions, and returned to the user. Even when encrypted in transit, this model concentrates risk at execution time and forces users to rely on policy and trust rather than technical enforcement.

Agentical.Net treats inference differently. By executing the inference locally, the platform collapses the trust boundary to the user's own environment. How inference is executed determines who controls the data, who owns the compute, and who ultimately governs the intelligence being deployed. For this reason, inference is not merely a technical detail but a strategic control point.

2.3.1 What is AI inference and why it matters

Inference is often oversimplified as merely "running a model." In reality, it is the operational stage of an AI system in which a trained model processes inputs to produce outputs. Training defines what a model is capable of in principle; inference determines how that ca-

pability is applied in practice. Every prompt, document, decision query, or automated workflow step goes through inference. This includes ingesting live inputs (prompts, documents, signals), gathering relevant context (retrieved knowledge, memory, state), performing the model's computations, and managing the resulting outputs and downstream effects.

In most modern AI systems, inference happens continuously and at scale. As AI moves from experimental use into persistent workflows, assistants, and autonomous agents, inference becomes a constant background operation rather than an occasional task. This makes the architecture of inference the most critical design decision in any AI platform. That being said, inference matters because:

- It processes the most sensitive data users provide.
- It dominates ongoing compute cost.
- It defines latency, reliability, and availability.
- It determines where trust must be placed.

Inference is the point at which AI systems interact most directly with sensitive data (personal information, confidential business material, or regulated content), and the retrieved context may expose internal knowledge bases or proprietary documents, the outputs may trigger actions or decisions with real consequences. Because of this, inference represents:

- **The highest privacy risk surface** — Live data is processed in real time, often without the opportunity for sanitization or aggregation.
- **The strongest trust dependency** — Users must trust that execution environments do not inspect, retain, or misuse their inputs.
- **The most consequential failure mode** — Errors or leaks at inference time affect real users and systems immediately.

Architecturally, this means that **inference placement determines whether an AI system can be safely integrated into sensitive or regulated environments**! In *Agentical's* case, the architecture reframes inference as a control point, meaning its:

- Controlled by local compute constraints.
- Governed by explicit coordination through the MCP layer.
- Augmented by private knowledge through local RAG.
- Extended through peer-to-peer communication only when intended.

In this model, inference becomes inspectable, composable, and governable — properties that are difficult to achieve when execution is remote.

2.4 Inference as a Sovereign Capability

Having established why AI inference is the most sensitive and consequential phase of AI execution, we can see what a central concept of the *Agentical.Net* architecture is: ***inference sovereignty***. Inference sovereignty refers to the ability of individuals and organizations to execute AI inference under their own control — on hardware they govern, within environments they trust, and according to rules they can technically enforce.

Why does sovereignty apply to inference? Because sovereignty is typically discussed in terms of data ownership or legal jurisdiction. However, in AI systems, control over inference is often more decisive than control over stored data. Even when data is nominally owned by the user, inference executed elsewhere can expose, transform, or act on that data in ways that escape effective oversight. Inference sovereignty matters because inference determines *1.* where data is actively processed, *2.* who can observe or influence execution, *3.* which rules apply at the moment of decision-making, and *4.* how outputs are generated and used

If inference is externalized, sovereignty is partial at best. True control requires that execution itself be local, inspectable, and bounded by user-defined constraints. In the context of *Agentical.Net*, inference sovereignty is not a political claim but a technical property. A system exhibits inference sovereignty when it satisfies the following conditions:

- **Local execution** — Inference runs on user-controlled hardware by default.
- **Explicit trust boundaries** — No implicit reliance on third-party execution environments.
- **Technical enforcement over policy reliance** — Guarantees are enforced by architecture, not terms of service.
- **User-governed data flow** — Inputs, context, and outputs move only when explicitly permitted.

These characteristics shift inference from a remote, managed service into a capability that runs directly within the user's own computing environment. As AI systems grow more autonomous and more tightly woven into sensitive workflows, control over where and how inference happens becomes increasingly critical. Because autonomous agents constantly handle context, memory, and private information, a centralized inference model leaves these systems in a state of ongoing dependence on external providers.

At this stage inference sovereignty is not just a privacy feature but a prerequisite for long-term resilience and autonomy. While privacy is a primary motivator, inference sovereignty has broader implications, especially:

1. **Operational resilience** — Local inference is not dependent on external availability, pricing changes, or service deprecation.
2. **Regulatory alignment** — Systems can be designed to comply with strict data residency and access requirements by construction.
3. **Strategic autonomy** — Organizations avoid hard dependencies on specific providers for core intelligence capabilities.
4. **Predictable economics** — Inference costs are tied to owned resources rather than variable consumption pricing.

These properties become increasingly important as AI systems move from experimentation into long-lived infrastructure. At this point, it is also important to understand why inference sovereignty may indeed be "the way of the future".

2.4.1 Why is inference sovereignty the way forward

Early AI systems were discretionary tools used occasionally, with limited scope and low integration into core workflows. In that context, outsourcing inference to centralized providers was acceptable. Today, AI is becoming embedded into knowledge work, internal automation and operational systems, regulated environments, and processes with institutional impact. As AI crosses this threshold, inference becomes part of the computing substrate and not an external service. Infrastructure-level components cannot rely on opaque execution environments without introducing systemic risk.

Inference sovereignty aligns AI execution with the expectations already applied to operating systems, databases, and networking stacks: control, inspectability, and predictable behavior.

The next important aspect is regulatory pressure. Data protection and AI governance frameworks increasingly focus on how data is processed, not merely where it is stored. Regulations such as GDPR, sector-specific compliance regimes, and emerging AI governance standards emphasize ***data minimization and purpose limitation***, ***explicit control over processing activities*** and ***auditability of automated decision-making***.

Centralized inference complicates compliance because execution occurs outside the user's control domain. Even when providers offer assurances, enforcement relies on contracts

and trust rather than architecture. Inference sovereignty resolves this tension structurally. When inference is local, many regulatory requirements are satisfied by design rather than by layered policy controls. This reduces compliance complexity and failure modes as systems evolve.

Another subject of inference sovereignty is concentration risk and systemic fragility, which means that small number of providers become chokepoints for *availability and continuity*, *pricing and economic access* and *policy enforcement and content control*. In short: **centralized inference creates concentration risk**.

As reliance on AI grows, these chokepoints become systemic vulnerabilities. Outages, service changes, or jurisdictional actions can have cascading effects across dependent systems.

Inference sovereignty distributes execution across user-controlled environments. While this reduces global optimization, it dramatically improves systemic resilience. Failures are localized rather than global, and users retain operational continuity independent of third-party service status.

The last subject of inference sovereignty that *Agentical.Net* sees is **changing dynamics of hardware usability**.

The persistence of centralized inference is increasingly disconnected from hardware reality. Modern consumer and professional devices now ship with GPUs capable of meaningful inference workloads, and standardized interfaces expose this capability safely to applications. As local compute becomes ubiquitous, continuing to externalize inference becomes a matter of convention rather than necessity. Inference sovereignty leverages existing hardware capacity instead of ignoring it, aligning system design with contemporary compute distribution.

Inference sovereignty offers an alternative path. By anchoring execution locally and treating models as interchangeable components, systems gain flexibility and long-term viability. This is especially critical for institutions expected to maintain continuity over decades rather than product cycles.

Agentical.Net positions inference as the next step in this evolution: **distributed, encrypted, and owned by the user**.

3 Agentical Use Cases

Agentical.Net is designed as a general-purpose AI platform whose architecture enables use cases that are difficult or impossible to support with centralized cloud-based systems. By combining local execution, encrypted peer-to-peer scaling, and private agents, *Agentical.Net* can be applied across personal, professional, and organizational contexts where privacy, control, and flexibility are critical.

3.1 Agentical.Net for Individual Users

For individual users, generative AI is increasingly integrated into personal knowledge work: writing, research, learning, planning, and decision support. In cloud-based systems, this typically requires exporting personal context — documents, notes, browsing history, or sensitive queries — to third-party services. Over time, this creates an asymmetry: the system becomes deeply informed about the user, while the user has little visibility into how that information is processed or retained.

In an individual context, *Agentical.Net* operates as a local AI environment running on user-controlled hardware and on the platform that is secure and does not require any programming knowledge, making use of AI technology as simple as possible.

For individual users, the most significant shift is that AI becomes an extension of the personal computing environment, not an external service. This has several consequences:

- Privacy guarantees are enforced technically, not contractually.
- AI availability is independent of subscriptions or service outages.
- Knowledge accumulation remains local and user-owned.

This model mirrors the evolution of personal computing itself: from shared mainframes to personal devices. *Agentical.Net* applies the same trajectory to AI inference and reasoning.

3.2 Agentical.Net for Teams & Collaboration

Many AI workflows are inherently collaborative. Teams use AI to synthesize shared knowledge, coordinate projects, review documents, and support collective decision-making. In these contexts, AI systems must operate across multiple users, devices, and trust boundaries while respecting internal access controls and data separation.

Traditional cloud-based AI platforms approach collaboration by centralizing execution and storage: team data is uploaded to a shared workspace, and inference is performed in a provider-managed environment. While this simplifies coordination, it collapses trust boundaries and forces teams to externalize sensitive internal context, making accountability and auditability to become critical. In centralized systems, these complexities are managed through platform-enforced access controls layered on top of shared infrastructure. In *Agentical.Net*, they are addressed through architectural separation: local execution, explicit permissions, and controlled communication.

In a team setting, each participant runs *Agentical.Net* locally, maintaining control over their own compute, data, and agents. **Collaboration emerges through explicit, encrypted peer-to-peer connections rather than shared cloud backends**. In this way, the system mechanisms of collaboration can achieve ***1.*** *peer-to-peer agent interaction where agents coordinate tasks or exchange results directly between authorized peers without central mediation*; ***2.*** *scoped knowledge sharing in which teams can share outputs without exposing underlying private data or full knowledge bases*; and ***3.*** *multi-step tasks can be distributed across agents and users, with MCP ensuring that each step operates within defined permissions and context boundaries.*

Because inference and retrieval occur locally, accountability is distributed rather than centralized, meaning we can inspect which agents ran, what data they accessed, and what outputs were shared. This improves traceability without introducing a central logging authority. In regulated or high-stakes environments, this local auditability is often preferable to opaque, provider-managed logs.

3.3 Agentical.Net for Enterprise & Regulated Sectors

Enterprises and regulated organizations — operating in finance, healthcare, legal services, government, and critical infrastructure — face constraints that fundamentally change how AI systems can be deployed. In these environments, data sensitivity, compliance obligations, auditability, and operational continuity are not secondary concerns; they are primary architectural requirements.

In regulated sectors, centralized AI platforms introduce several structural challenges:

1. Unclear data residency and jurisdictional exposure — Inference executed on external infrastructure may fall under foreign legal regimes, regardless of contractual terms.
2. Limited auditability of execution — Organizations must often rely on provider logs and assurances rather than direct technical verification.
3. Shared infrastructure risk — Multi-tenant environments increase exposure to misconfiguration, cross-tenant leakage, and systemic outages.
4. Compliance through policy rather than

architecture — Controls are often layered on top of systems that were not designed for strict governance from the outset.

These issues are difficult to remediate incrementally because they stem from where inference occurs and who controls it. *Agentical.Net* addresses these enterprise and regulatory requirements by shifting control inward, rather than attempting to harden external systems. This means that architectural properties of the platform includes:

Local and on-premise inference: AI execution occurs on infrastructure owned or directly governed by the organization — workstations, secure environments, or on-premise systems — eliminating the need to export sensitive data for inference.

Explicit trust boundaries: Data, knowledge, and execution contexts are isolated by design, reducing reliance on provider-side access controls.

Decentralized collaboration: Teams and departments can coordinate via encrypted peer-to-peer communication without creating centralized data aggregation points.

Agent governance: The MCP layer enforces explicit workflows, permissions, and memory scopes, enabling controlled automation without autonomous overreach.

One of *Agentical.Net's* core advantages in regulated environments is that many compliance requirements are satisfied architecturally, rather than procedurally. For example in the case of data minimization which **is enforced because inference and retrieval occur locally**, or in a case of data residency which **is guaranteed by execution locality rather than contractual promises**. This reduces the operational burden of compliance and lowers the risk of unintentional violations as systems evolve.

Another key element in providing suitable AI system for enterprises are so called deployment models. *Agentical.Net* supports multiple enterprise deployment patterns without changing its core architecture. The models can be **fully on-premise environments** (suitable for highly regulated or air-gapped systems), **hybrid enterprise setups** (local inference with optional peer-to-peer collaboration across organizational boundaries), and **edge and restricted networks** (operation in environments with limited or intermittent connectivity. Because *Agentical.Net* does not depend on centralized inference services, these deployment models differ primarily in policy and configuration, not in fundamental system behavior.

3.4 Agentical.Net for Professionals

Professionals such as lawyers, consultants, researchers, engineers, analysts, designers, and clinicians increasingly rely on AI to augment complex, high-value work. These workflows are typically characterized by confidential data, domain expertise, and accountability for outcomes. Unlike large enterprises, professionals often operate without dedicated IT or compliance teams, yet they face equally stringent requirements around privacy, trust, and correctness.

Professional contexts differ from both individual and enterprise use in several key ways:

1. High sensitivity, low tolerance for leakage (client materials, research notes, drafts, and analyses are often confidential by default).

2. Deep, iterative workflows (work is rarely single-turn; it involves sustained reasoning, revision, and contextual memory.

3. Personal accountability (outputs are attributed to the professional, not the tool, increasing the need for transparency and control.

Cloud-based AI systems struggle in this space because they require professionals to externalize the very context that defines their expertise, but in the case of *Agentical.Net* we enable professionals to integrate AI into their practice as a local cognitive tool. That means that:

- Drafts, case files, research data, or proprietary analyses can be processed without leaving the professional's environment.
- Personal knowledge bases (notes, archives, prior work) can be indexed locally and used for contextual reasoning without uploading them to shared platforms.
- MCP-coordinated agents can manage multi-step tasks such as document review, comparative analysis, literature synthesis, or project planning.
- Results can be shared with clients or collaborators in scoped form, without exposing raw materials or internal reasoning.

Professionals often develop repeatable workflows (checklists, analyses, or reasoning patterns) that represent valuable expertise. Through *Agentical.Net's* agent sharing layer and monetization mechanisms, these workflows can be encapsulated as agents and reused across clients or project, shared with peers or teams, and monetized without exposing client data or execution context. This allows professionals to scale their expertise while preserving confidentiality and control.

3.5 Legal, Ethical & Governance Considerations

As generative AI systems move into routine use across sensitive and regulated domains, legal, ethical, and governance considerations become inseparable from system design. These considerations are often treated as external constraints addressed through policies, contracts, or post hoc controls. At *Agentical.Net*, they are treated as architectural requirements that shape how AI systems are built and operated.

Many of the most challenging governance failures in AI systems arise not from malicious intent, but from mismatches between policy and architecture. When execution is centralized and opaque, compliance depends on assurances that are difficult to verify and easy to break as systems evolve. There are some key governance risks in conventional AI platforms that include:

- Inability to verify how and where inference occurs.
- Over-collection or unintended retention of data.
- Ambiguous responsibility for automated decisions.
- Limited auditability of model behavior in context.

Agentical.Net addresses these risks by aligning legal and ethical principles with technical enforcement points: **where inference runs, how data flows, and who controls execution.**

From a legal perspective, many regulatory obligations focus on processing, not just storage. Requirements around data minimization, purpose limitation, and lawful access are significantly easier to satisfy when data does not leave the user's environment in the first place. We believe that the architecture has to support legal alignment and with that it is important that *Agentical.Net* encompass:

- **Local inference by default**, thus reducing cross-border data transfer concerns.
- **Limited data exposure** to task-scoped, ephemeral contexts.
- **Avoidance of centralized retention** of prompts, embeddings, or outputs.
- **Allowing organizations and individuals to define their own compliance posture** through configuration rather than negotiation with providers.

This does not eliminate legal responsibility, but it simplifies it. Compliance becomes a property of system deployment rather than a dependency on third-party assurances.

Legal, ethical, and governance concerns are not peripheral to AI systems but they define whether those systems can be trusted,

adopted, and sustained over time. *Agentical.Net* addresses these concerns by embedding governance into the structure of AI execution itself. By aligning legal and ethical principles with technical design, we enable AI systems that are not only easy-to-use and powerful, but governable by construction — supporting responsible use across personal, professional, and regulated environments.

3.5.1 Data ownership & user rights

Data ownership and user rights are foundational concerns in any AI system, but they become particularly acute when inference operates on live, private, or regulated information. In many contemporary AI platforms, ownership is defined contractually while control is exercised technically by the provider. This disconnect creates ambiguity around who truly governs data usage, retention, and secondary processing. In centralized AI systems, users may nominally "own" their data, but inference execution occurs on infrastructure they do not control. As a result:

- Prompts and inputs may be logged or retained transiently.
- Derived artifacts (embeddings, telemetry, metadata) may persist beyond user intent.
- Data may be subject to provider-side access, analysis, or legal compulsion.
- Users have limited ability to verify how their data is processed.

This creates a gap between **formal ownership** and **effective control**. From a governance perspective, user rights that cannot be technically enforced are inherently fragile.

Agentical.Net resolves this gap by making data ownership operational rather than declarative. Ownership is reflected in **where data resides, where it is processed, and who can authorize its movement**. The key properties include:

- **Local data residency by default** — User data and outputs remains on user-controlled hardware unless explicitly shared.
- **No implicit data capture or retention** — *Agentical.Net* does not require centralized logging, storage, or telemetry of user data to function.
- **User-governed data flow** — Data moves only through explicit actions: sharing via peer-to-peer communication, exporting artifacts, or granting scoped agent permissions.

In this model, ownership is enforced structurally: if data never leaves the user's environment, it cannot be appropriated, repurposed, or accessed without consent.

User Rights by Design: Agentical's architecture supports core user rights commonly articulated in data protection and AI governance frameworks (see Table 1). These rights are not implemented as optional features, but as consequences of local-first execution and explicit coordination.

A frequent blind spot in AI governance is **derived data**. In many platforms, these artifacts are treated as non-user data and retained indefinitely. *Agentical.Net* treats derived data as **subject to the same ownership and control expectations as source data**. This prevents secondary use or accumulation of user-derived signals outside the user's control.

4 Competitive Landscape

Most AI platforms optimize around one of two poles: *centralized inference for scale and convenience*, or *local execution for control and privacy. Agentical.Net* differs in that it treats

Governance Principle	Architectural Requirement	Agentical Enforcement Mechanism
Data Ownership	Data must remain under user control	Local data residency; no centralized storage; explicit peer-to-peer sharing only
User Consent	Data access must be intentional and scoped	MCP-declared permissions; opt-in knowledge ingestion; explicit agent capabilities
Data Minimization	Process only what is required for the task	Task-scoped context assembly; ephemeral RAG retrieval; no background data collection
Right to Access	Users must be able to inspect data use	Local observability of agents, knowledge sources, and inference context
Right to Revocation	Access must be withdrawable at any time	Local permission controls; revocable agent memory and knowledge bindings
Right to Deletion	Data and derivatives must be removable	Local storage of documents, embeddings, and indices; verifiable deletion
Purpose Limitation	Data used only for declared intent	MCP-governed workflows; separation of inference, retrieval, and memory
Auditability	Execution must be traceable	Local execution logs; inspectable agent workflows; no opaque provider log
Jurisdictional Compliance	Processing location must be controllable	On-device / on-premise inference; no mandatory cross-border execution
Accountability	Responsibility must be clearly attributable	Local execution environments; explicit agent ownership and configuration

Table 1: Legal, ethical, and governance mechanisms

local inference, peer-to-peer communication, and agent coordination as first-class architectural primitives that are designed to work together. These choices place *Agentical.Net* in a distinct category: **NOT** a hosted AI service, **NOT** a local runner, and **NOT** a decentralized compute marketplace, but an **execution and coordination layer for sovereign AI workflows**.

4.1 Architectural Axes of Comparison & Agentical's Divergence

Across the AI ecosystem, platforms primarily differ along four architectural axes:

1. **Inference Location** — where execution physically occurs
2. **Knowledge Handling** — how private data is accessed and retained
3. **Coordination Model** — how multi-step workflows and collaboration are managed
4. **Trust Enforcement** — whether guarantees are architectural or contractual

These axes determine not only system behavior, but also long-term viability in sensitive or regulated environments. At *Agentical.Net* the distinguishing feature is that we optimize *all four axes simultaneously* for local control, rather than optimizing one at the expense of the others.

Agentical.Net vs. Cloud AI: Cloud AI platforms centralize model hosting, inference, and often knowledge management. This model excels at elasticity and managed convenience, but it externalizes execution and data

by default (see Table 2). *Agentical.Net* does not attempt to replace cloud AI for all workloads. Instead, we target use cases where control over inference and data is non-negotiable.

Agentical.Net vs. Local Runners (Ollama, LM Studio): Local runners enable on-device model execution and represent an important step toward inference locality. However, they typically focus on single-user execution and lack system-level coordination primitives. *Agentical.Net* builds on the premise of local execution but extends it into a multi-agent, collaborative system, with explicit boundaries around knowledge access, memory, and sharing.

Agentical.Net vs. NotebookLM: Notebook-style tools integrate documents, notes, and AI reasoning into a single interface. They are effective for individual synthesis and exploration but rely on centralized execution and storage. *Agentical.Net* offers a different approach: notebook-like reasoning can be achieved without centralizing documents or inference, using local RAG and agent workflows that remain under user control.

Agentical.Net vs. Decentralized Compute Networks: Decentralized compute networks aim to distribute inference across many providers using market-based allocation. While they reduce dependence on single vendors, they still externalize execution to third-party hardware. *Agentical.Net* prioritizes sovereignty over decentralization for its own sake. Execution occurs on hardware the user governs, and peer-to-peer networking is used for coordination — not outsourcing inference.

Dimentions	**Cloud AI**	**Local Runners**	**Notebook Tools**	**Decentralized Nets**	**Agentical.Net**
Inference Location	Centralized	Local	Centralized	External	Local
Knowledge Handling	Uploaded	Manual	Uploaded	External	Local RAG
Coordination	Provider	None	Document-based	Market-based	MCP Agents
Collaboration	Cloud workspaces	N/A	Shared docs	Network level	Secure P2P
Trust Model	Contractual	Implicit	Contractual	Economic	Architectural
Governance	Policy-layer	User-managed	Provider-managed	Weak	By design

Table 2: Comparative architectural summary

Our architecture is intentionally constrained to serve environments where **control**, **auditability**, and **data sovereignty are first-order requirements**. With this we pivot in strategic positioning that has its tradeoffs, mostly in reduced elastic scale —in exchange for execution control —; explicit configuration — in exchange for transparency —; and local responsibility — in exchange for enforceable rights. These tradeoffs are unacceptable in some contexts and essential in others.

The competitive landscape for AI is best understood as a set of architectural choices about where inference runs and who governs it. *Agentical.Net* occupies a distinct position by integrating local inference, private knowledge access, agent coordination, and peer-to-peer collaboration into a single coherent system. This positioning reflects a long-term view: *as AI becomes infrastructure rather than a service, execution locality and governance by design become competitive advantages rather than limitations.*

ROADMAP

TBD

under construction

> milestones and timeline will be defined here

References

[1] Austin Eng, François Beaufort, Deepti Gandluri. Webassembly and webgpu enhancements for faster web ai. https://developer.chrome.com/blog/io24-webassembly-webgpu-1, 2024.

[2] Cloud security alliance. Top threats to cloud computing 2024. https://cloudsecurityalliance.org/artifacts/top-threats-to-cloud-computing-2024, 2024.

[3] European Commision. Artificial intelligence – questions and answers. https://ec.europa.eu/commission/presscorner/detail/en/qanda_21_1683, 2024.

[4] European Union agency for cybersecurity. Enisa threat landscape. https://www.enisa.europa.eu/sites/default/files/publications/ENISA%20Threat%20Landscape%202023.pdf, 2023.

[5] National Institute of Standards and Technology. Artificial intelligence risk management-framework (ai rmf 1.0). https://doi.org/10.6028/NIST.AI.100-1, 2023. Accessed: 2026-02-02.

[6] W3C. Webgpu. https://www.w3.org/TR/webgpu/, 2025.

[7] Yang Gu. Unlock the potential of ai and immersive web applications with webgpu. https://www.intel.com/content/www/us/en/developer/articles/technical/unlock-potential-ai-immersive-web-apps-with-webgpu.html, 2024.