Briefing Note: Using Local-Only AI in a Micro-Enterprise

Author: Alec Fearon, 23 November 2025

Transparency label: AI-assisted

This note was drafted collaboratively. Alec defined the purpose, scope, and required level of explanation. ChatGPT generated the prose, refined the structure, and made several rounds of targeted improvements within a clear control framework. Alec provided direction, corrections, and quality checks at each stage. The final document reflects human oversight and editorial judgement, supported by AI drafting and revision.

This note outlines a practical way for a micro-enterprise to run AI entirely on local machines. It treats the three-layer context model as the human-facing control framework, and places a small set of lightweight, local models underneath it. The aim is to describe a workable pattern, not mandate an architecture.

1. A control framework that sits above the technology

The three-layer model governs how staff interact with the AI. It remains stable whether the models run locally or in the cloud.

Business context describes what the organisation does, who it serves, and what constraints matter. It is the AI's operating profile and should be stored as versioned configuration.

The governance layer combines written rules ("don't invent facts; ask when uncertain") with technical limits enforced by software or operating system permissions. These constraints sit outside the model and prevent it from accessing tools or information it shouldn't.

Task prompts are predefined workflows: extract a booking, draft a message, summarise a policy, answer a question using internal documents. Staff select the task rather than free-typing instructions. This reduces ambiguity and keeps behaviour predictable.

These three layers define the organisation's intent for how the AI should behave. That intent is realised only when the orchestrator applies the layers consistently and in the right order.

2. What the local technology stack looks like

Running everything locally works best when different jobs are handled by different models rather than forcing one model to do everything.

A practical local stack contains:

1. A classifier / extractor model (1–4B parameters). Fast and inexpensive. Ideal for structured extraction tasks such as pulling booking details from emails. Runs well on CPU.

2. A drafter / rewriter model (7–8B parameters). Used for writing clear emails, summaries, and internal text. With the business context and governance layer wrapped around it, this model behaves like a competent administrative assistant.

3. A retrieval-augmented generation (RAG) pipeline for search and reasoning. This is a process rather than a separate model. It retrieves relevant text from your documents, and the drafter model answers using only that material. Long-context 7–8B models handle this reliably.

This three-part stack covers most real cases a micro-enterprise will face. Each component has a distinct role, which keeps the system simple, transparent, and easy to maintain.

3. No need for fine-tuning

Fine-tuning a model is brittle, costly, and unnecessary. The models above can be used "out of the box"; they do not need fine-tuning unless the organisation has large domain-specific datasets. For a micro-enterprise, improvement should focus on refining the control framework and strengthening the orchestrator rather than altering model weights. Aim for:

- maintain a clear and stable business context,
- clean and enforced governance rules,
- well-designed task prompts,
- and tidy internal documents.

These elements have more impact on quality and reliability than adjusting model weights.

4. How the RAG pipeline fits in

Retrieval-augmented generation keeps answers tied to the organisation’s actual content. It has two loops.

Ingestion: documents are split into chunks, embedded, and stored in a small local index. This loop runs offline whenever content is added or updated.

Query: the orchestrator retrieves the most relevant chunks and constructs a prompt that instructs the model to answer using only those excerpts. This loop runs in real time when a user asks a question.

This ensures accuracy and traceability. It also reinforces governance: the model is explicitly instructed to answer only from retrieved excerpts, not from its general training.

5. Orchestration

Users of this local-only AI system work with it through an orchestrator. The orchestrator is not an AI. It is a small, traditional piece of software whose behaviour, unlike that of an AI, is deterministic. The orchestrator controls all parts of the technology stack; it:

- chooses which model or tool to call for **any** task,

- assembles prompts from the three layers of the control framework and, where relevant, retrieved content,
- enforces guardrails and permissions across the entire system,
- and logs each request for accountability.

The orchestrator is needed because local models cannot manage themselves. They do not know what task they are being used for, which documents they may access, or which rules apply. Without an orchestrator, any model would simply process whatever text it receives, with no way to:

- ensure the appropriate model is chosen for the task,
- combine business context, governance, and task instructions reliably,
- enforce permissions or safety limits,
- or ensure RAG answers are grounded in retrieved documents.

The orchestrator provides this structure. It coordinates all AI-assisted work and ensures the system behaves consistently and safely.

To create an orchestrator we need a backend engineer who understands how prompts, retrieval, and limits shape AI behaviour. This person is a software engineer competent in building the parts of a system that live behind the scenes; they work on the logic, APIs, data handling, and services that support an application, not the visual interface.

6. Why this approach works for micro-enterprises

A local-only policy imposes limits on model size, but not on usefulness. The combination of:

- a clear human-facing control framework,
- a small set of specialised local models,
- disciplined document management,
- and a simple orchestrator,

creates a system that is private, predictable, inexpensive, and easy to expand in small steps.

The core advantage is control. Local-only AI behaves well when the organisation defines clear boundaries through the three-layer control framework and uses lightweight technology to enforce those boundaries in practice.