ERPNext AI Agent

Product Architecture and Implementation Plan

Native Frappe app | MCP-first tool layer | Public and private LLM | Desk, WhatsApp, Telegram, MCP clients

What this document is

A complete plan for building and selling an AI Agent for ERPNext: the decision between architecture options, every layer of the recommended design, the security and identity model for web and mobile channels, the data model, the technology stack, deployment topologies for startups and enterprises, a phased roadmap, team, cost drivers and risks.

How to read it

Sections 1 to 3 give the decision and the reasoning. Sections 4 to 14 describe each layer in detail with diagrams. Sections 15 to 22 cover security, data model, stack, roadmap, packaging and risks. Appendices hold the tool catalogue, settings reference and glossary.

Decision in one sentence

Build a native Frappe app whose core is a permission-aware, audited tool layer exposed through MCP, with the chat experience delivered inside ERPNext Desk and through WhatsApp and Telegram via a channel gateway, and with public or private LLMs selected by configuration.

Contents

1. Executive summary

Three approaches were evaluated. Approach A, where users paste an ERPNext API key and secret into Claude or ChatGPT, fails enterprise security review because the key is a permanent full-permission credential stored outside the customer's control, with no scoping, no approval gate and no distinguishable audit trail. Approach B, a native Frappe app with an embedded chat, is the correct core: the backend acts as the logged-in user, keys never leave the server, every action is permission-checked and audited. Approach C, private LLMs, is not a separate architecture; it is a provider setting inside B plus a local embedding model and a deployment guide.

Mobile access changes the design more than adding a client. WhatsApp and Telegram messages arrive with no Frappe session, so the system needs an identity binding, a durable approval mechanism, channel-aware policy and a format adapter. The user never types credentials into a chat app; they type at most the site URL and complete linking by logging in to their own ERPNext in a browser, after which the gateway holds only a scoped, revocable OAuth token.

Design principles

- **MCP-first.** The product is the tool layer and the policy around it. The chat window is the first client, not the product.
- **Act as the user, always.** Every read and write goes through the Frappe ORM as the real user. Never bypass permissions.
- **No credential ever leaves the server.** LLM keys stay in AI Settings. ERP credentials are never typed into a chat.
- **Risk-tiered actions with human approval.** Reads run freely, drafts are visible, submits need confirmation, destructive actions are off by default.
- **Everything is audited.** Prompt, tool, arguments, decision, approver, channel and linked document.
- **Provider-agnostic.** Public and private models switch by configuration, each shipped as a tested profile.
- **Channel-agnostic.** Desk, WhatsApp, Telegram, self-hosted messengers and external MCP clients share one agent.

Headline numbers

Item	Value
Delivery to general availability	About 25 weeks across four phases
Core team	4 full-time engineers plus part-time DevOps, QA and product
Initial tool catalogue	18 tools covering sales, purchase, stock, accounts, HR and reports
Private model minimum	One 24 GB GPU for a 30B-class model; two 80 GB GPUs for 70B-class
Commercial tiers	Startup (BYOK public), Business (metered), Enterprise (private, own channels)

2. Goals, scope and assumptions

Goals

- Let ERPNext users ask questions and perform actions in natural language: create Sales Orders, check stock, run reports, create invoices, query any DocType, summarise ledgers, handle HR requests.
- Work from the ERPNext Desk and from a mobile phone through WhatsApp and Telegram, with voice notes and photos.
- Enforce the user's existing ERPNext permissions so the agent can never do more than the user could do by hand.
- Keep a complete audit trail acceptable to finance auditors and security teams.
- Support public LLMs for startups and fully private LLMs for enterprises with one codebase.
- Be sellable as a product to many ERPNext customers, self-hosted or hosted by the vendor.

Non-goals for version 1

- Autonomous background agents that act without a human request. Proactive alerts come in phase 4 and are notifications, not actions.
- Fine-tuning models on customer data. Retrieval and tools cover the need at far lower risk.
- Replacing ERPNext forms. The agent drafts and proposes; the forms remain the source of truth.
- A native iOS or Android app. The Desk progressive web app and messaging channels cover mobile.

Assumptions

- Customers run Frappe v15 or later on MariaDB with Redis and background workers, the standard bench layout.
- Customers are mostly in Saudi Arabia and the Gulf, so Arabic and English must both work in chat, voice and retrieval.
- WhatsApp is the dominant mobile channel; Telegram is cheaper to start with and used for development first.
- Enterprise customers may forbid public LLMs and may also forbid WhatsApp for sensitive data. Both must be policy settings.
- Saudi PDPL and similar regulations apply: data residency, purpose limitation and auditability matter.

3. Approach comparison: A, B and C

Criterion	A: API key pasted into Claude / ChatGPT	B: Native Frappe app, embedded chat	C: Private LLM
Security	Weak. Key equals full user permissions, stored by a third party, never expires, cannot be scoped. Prompt injection can trigger writes with no approval gate.	Strong. Runs as the logged-in user. Keys never leave the server. Every tool call is permission-checked and can require confirmation.	Strongest for residency. Same as B plus no data egress.
Permissions	Whatever the key holder can do.	Frappe role, document and field permissions, plus an AI policy layer.	Same as B.
Audit	Only ERPNext Version logs, which cannot separate AI from human actions.	Full conversation, tool, decision, approver and linked document.	Same as B.
Rate limits and cost	Uncontrolled.	Per-user and per-company budgets. BYOK or metered.	Fixed GPU cost, no per-token cost.
User experience	Good chat, no ERP context, window switching.	Best for daily ERP work: knows the open form and filters.	Same as B; local models are slower and weaker unless 30B or larger.
Scalability	Vendor dependent.	Standard Frappe scaling with workers and Redis; LLM calls are async jobs.	Needs vLLM and GPUs. Ollama for pilots only.
Future-proofing	Being replaced by OAuth-based remote MCP connectors.	MCP-native backend plugs into every new client and agent gateway.	New models drop in by configuration.
Enterprise adoption	Blocked by security teams.	Passes standard review.	Required by regulated buyers.
Build effort	Near zero.	Medium.	B plus roughly two weeks.

Verdict. Build B. Fold C into B as the provider layer. Deliver the legitimate version of A, external clients such as Claude and ChatGPT, by exposing the same tool layer as a remote MCP server protected by OAuth 2.1, so those clients never hold an ERPNext credential.

Inside or outside ERPNext: what wins long term

Inside wins for the daily user because the agent has screen context, the user's session and one-click confirmations. Outside wins for cross-system work, where a manager in Claude asks about ERPNext, email and CRM together. Enterprises are moving toward agent gateways that speak MCP. Building MCP-first means the product wins in both places and does not depend on any single chat vendor.

4. Architecture overview

The system is organised in ten layers. Requests enter from a client, pass through the channel gateway and identity layer, are handled by the agent runtime, checked by the policy engine, executed by tools under the user's permissions, and every step is recorded by the audit layer. The LLM provider and knowledge layers are services used by the runtime.

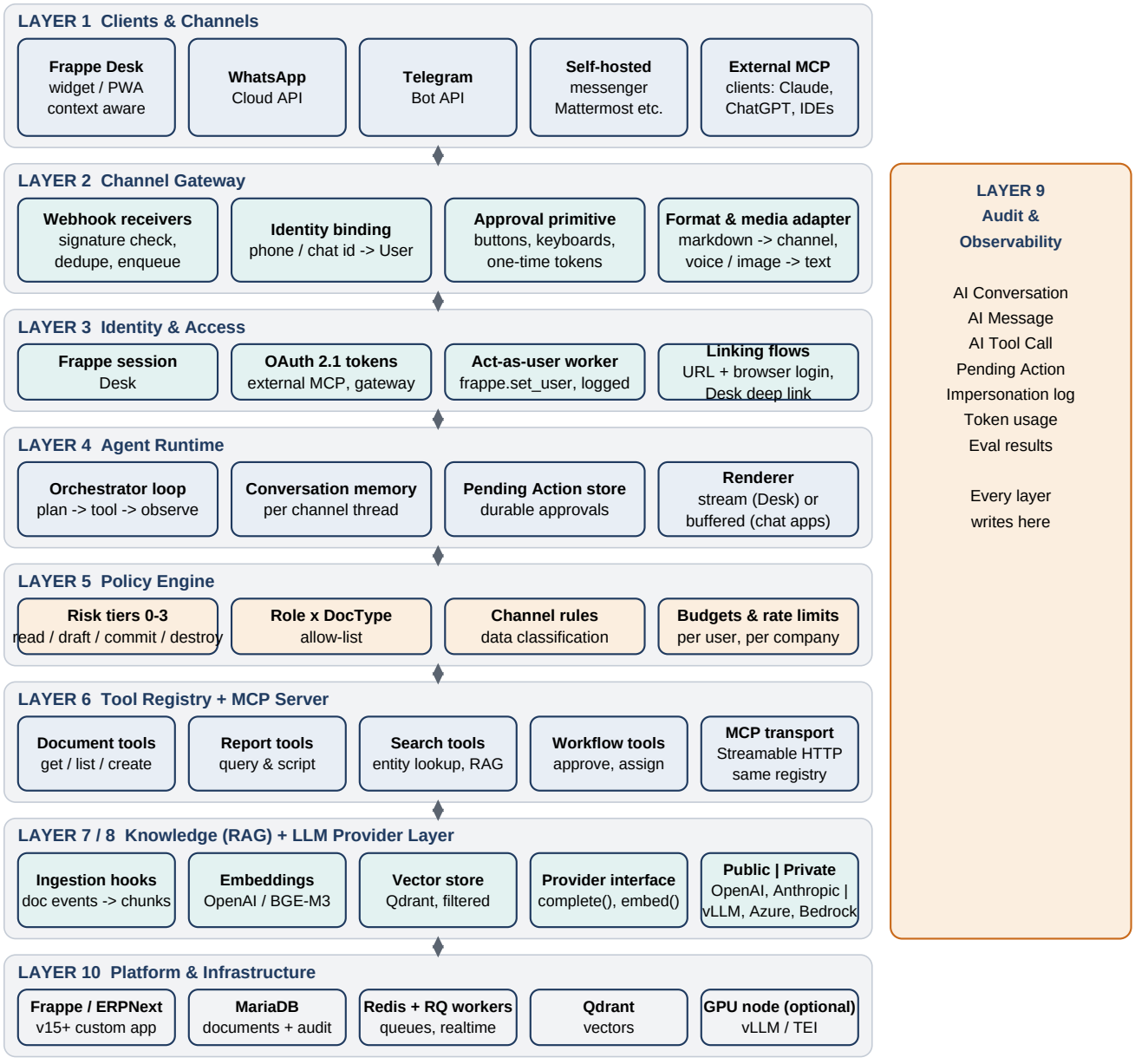


Figure 1. Layered architecture of the erpnext_ai_agent app. Layer 9, audit, receives events from every layer.

Layer	Responsibility	Section
1 Clients and channels	Where users talk to the agent: Desk widget and PWA, WhatsApp, Telegram, self-hosted messenger, external MCP clients	5
2 Channel gateway	Receive and verify webhooks, bind identities, render approvals, adapt formats and media	6
3 Identity and access	Sessions, OAuth tokens, act-as-user worker, linking flows	7
4 Agent runtime	Orchestration loop, memory, pending actions, streaming or buffered output	8
5 Policy engine	Risk tiers, role and DocType allow-lists, channel rules, budgets	9

Layer	Responsibility	Section
6 Tool registry and MCP	Plain Python tools over the Frappe ORM, exposed in-process and over MCP	10
7 Knowledge and RAG	Ingestion, embeddings, permission-filtered retrieval	11
8 LLM provider layer	One interface, public and private drivers, model profiles	12
9 Audit and observability	Conversation, message, tool call, pending action, usage, evals	13
10 Platform	Frappe, MariaDB, Redis, workers, Qdrant, optional GPU node	14

5. Layer 1: Clients and channels

Client	Identity source	Context available	Approval UI	Notes
Frappe Desk chat widget	Frappe session cookie and CSRF token	Current route, open DocType and record, list filters, selected rows	Confirm and Cancel buttons in the chat, diff preview of the proposed document	Streams tokens over Socket.IO. Also installs as a PWA on phones.
WhatsApp Cloud API	Phone number bound to a Frappe user	None; agent must ask more clarifying questions	Reply buttons up to three, list messages, optional PIN, or a link into Desk	Meta business verification per customer, 24-hour service window, templates for proactive messages.
Telegram Bot API	Telegram user id bound to a Frappe user	None	Inline keyboard buttons with callback data	Each customer creates their own bot; simplest first connector.
Self-hosted messenger (Mattermost, Rocket.Chat)	Messenger account bound to a Frappe user, or SSO	None	Interactive message buttons	The private mobile channel for customers who forbid WhatsApp.
External MCP clients (Claude, ChatGPT, Copilot Studio, IDEs)	OAuth 2.1 token issued by the customer's Frappe site	Whatever the client passes	Client's own tool approval prompts plus server-side pending action	Replaces approach A. Same tools, same policy.
Slack and voice (phase 4)	Slack user id or phone number bound to a Frappe user	None	Slack block buttons, spoken confirmation plus PIN	Voice uses realtime speech APIs publicly or Whisper privately.

Desk widget behaviour

- Lives as a floating panel on every Desk page and as a full page at /app/ai-agent for longer sessions.
- Sends the current route and document name with each message so the agent can answer "summarise this invoice" without asking.
- Shows proposed documents as a preview card with field values before the user confirms.
- Opens created documents in a new tab with one click and links back to the conversation from the document's timeline.
- Supports Arabic right-to-left rendering and voice input through the browser's speech API.

Mobile behaviour rules

- Short answers. Tables become bullet lists. Reports over a size limit become a PDF attachment or a deep link into Desk.
- Voice notes are transcribed before reaching the agent; the transcript is shown back to the user for correction when confidence is low.
- Photos of invoices or receipts go through a vision model to extract fields, then the normal create-draft tool runs.
- The agent confirms entity matches explicitly on mobile, for example which of two similarly named customers is meant.

6. Layer 2: Channel gateway

The gateway is the only code that knows about a specific messaging platform. Every channel implements one small interface so the agent runtime never changes when a channel is added.

```
receive(request) -> InboundMessage | verify_signature(request) | resolve_identity(external_id) ->
User | send_text(user, text) | send_approval(user, pending_action) | send_file(user, file) |
send_typing(user)
```

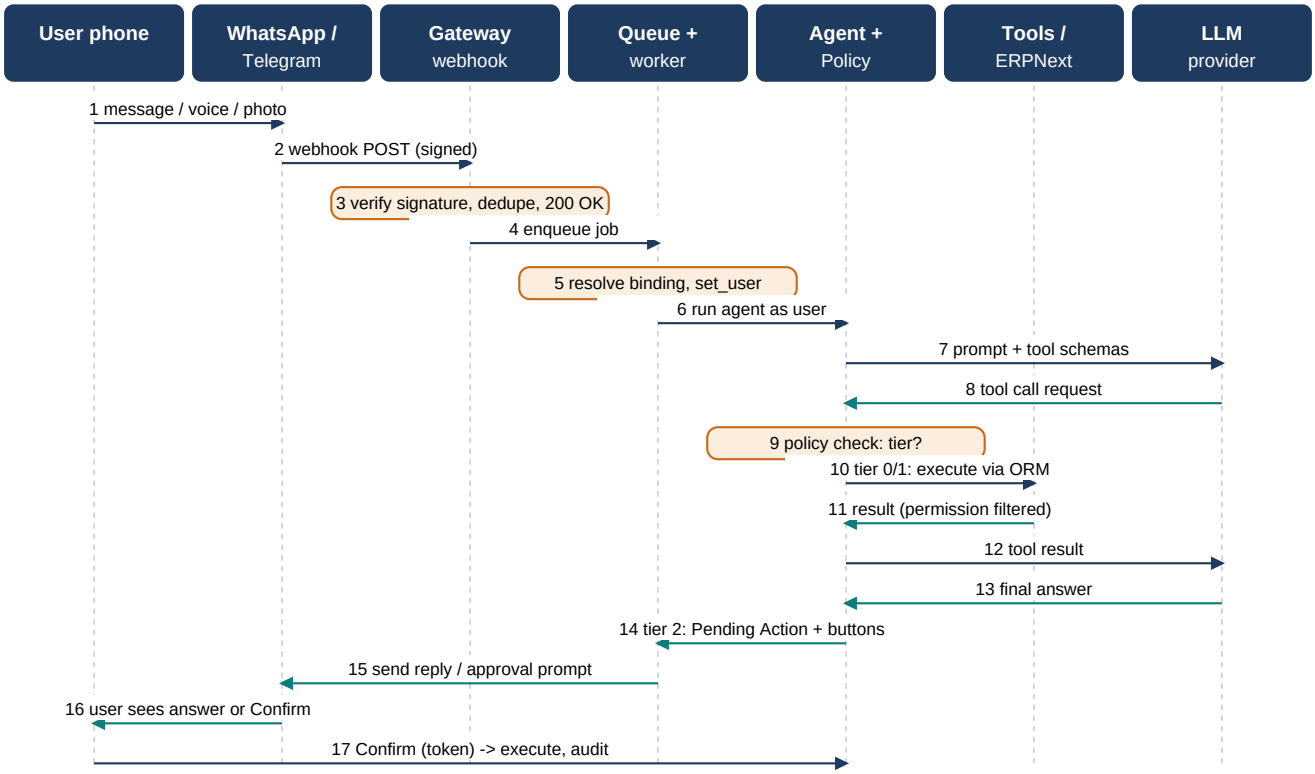


Figure 2. End-to-end flow of one WhatsApp message, including the deferred confirmation of a tier 2 action.

Responsibilities

Component	Detail
Webhook receiver	Guest-accessible whitelisted endpoint per channel. Verifies the platform signature (X-Hub-Signature-256 for WhatsApp, secret token header for Telegram), deduplicates on message id, returns 200 immediately and enqueues a background job. Never runs the agent inline.
Identity resolution	Looks up Channel Identity by channel and external id. Unbound senders receive only linking instructions and never reach the LLM.
Approval primitive	Renders a Pending AI Action as platform buttons carrying a single-use token. Handles the callback, validates token, expiry and PIN if required, and hands off to the runtime for execution.
Format adapter	Converts the runtime's markdown result to platform formatting, splits long messages, converts tables to lists, attaches files.
Media adapter	Downloads voice notes and images from the platform, runs transcription or vision through the provider layer, and passes text plus the original file reference to the runtime.
Outbound queue	Rate-limited sender with retries, respecting the WhatsApp 24-hour window and using approved templates outside it.
Tenant routing (shared gateway only)	Maps phone hash to site URL, user and token; forwards the message to that site's agent endpoint over HTTPS with the token.

Platform specifics to plan for

- **WhatsApp.** Meta Business verification per customer, a dedicated number per customer, no shared numbers across tenants. A Business Solution Provider such as 360dialog or Twilio simplifies onboarding. Webhooks retry on slow responses. Free-form replies are allowed for 24 hours after the user's last message; proactive alerts outside that window need pre-approved templates.
- **Telegram.** Each customer creates a bot and stores the token in AI Settings. Use the secret token header on the webhook. Start parameters support deep-link onboarding. Free and fast, which is why it is the first connector built.
- **Privacy.** Both platforms relay messages through their own servers. The private tier must be able to disable them by policy and offer the Desk PWA or a self-hosted messenger instead.

7. Layer 3: Identity, linking and access

Three ways a request proves who it is. In Desk, the normal Frappe session is used and nothing extra is needed. For external MCP clients and the shared gateway, an OAuth 2.1 token issued by the customer's own Frappe site is used; Frappe ships an OAuth 2 provider, so no new authentication system is built. For messaging channels, a verified binding between the platform identity and a Frappe user is used, and a background worker acts as that user.

Why credentials are never collected in chat

Asking a mobile user for the ERPNext URL, API key and API secret inside WhatsApp or Telegram is rejected as a design, including as a fallback. A fallback that exists becomes the path most users take.

Aspect	URL plus API key and secret typed in chat	URL in chat plus browser login (chosen)
Credential exposure	Permanent secret in chat history, phone backups, linked desktop clients and the platform's servers.	Scoped token stored encrypted on the server; nothing sensitive in the chat.
Revocation	Must regenerate keys in ERPNext, which breaks other integrations.	One click in Desk unlinks the number.
Expiry	None.	Refreshable, expiring tokens.
Lost or shared phone	Full account compromise.	Attacker gets the chat; the binding is revoked from Desk.
Multi-tenant support	Yes.	Yes.
User effort	Copy two long strings from ERPNext settings.	Tap a link and log in.
Enterprise acceptance	No.	Yes.

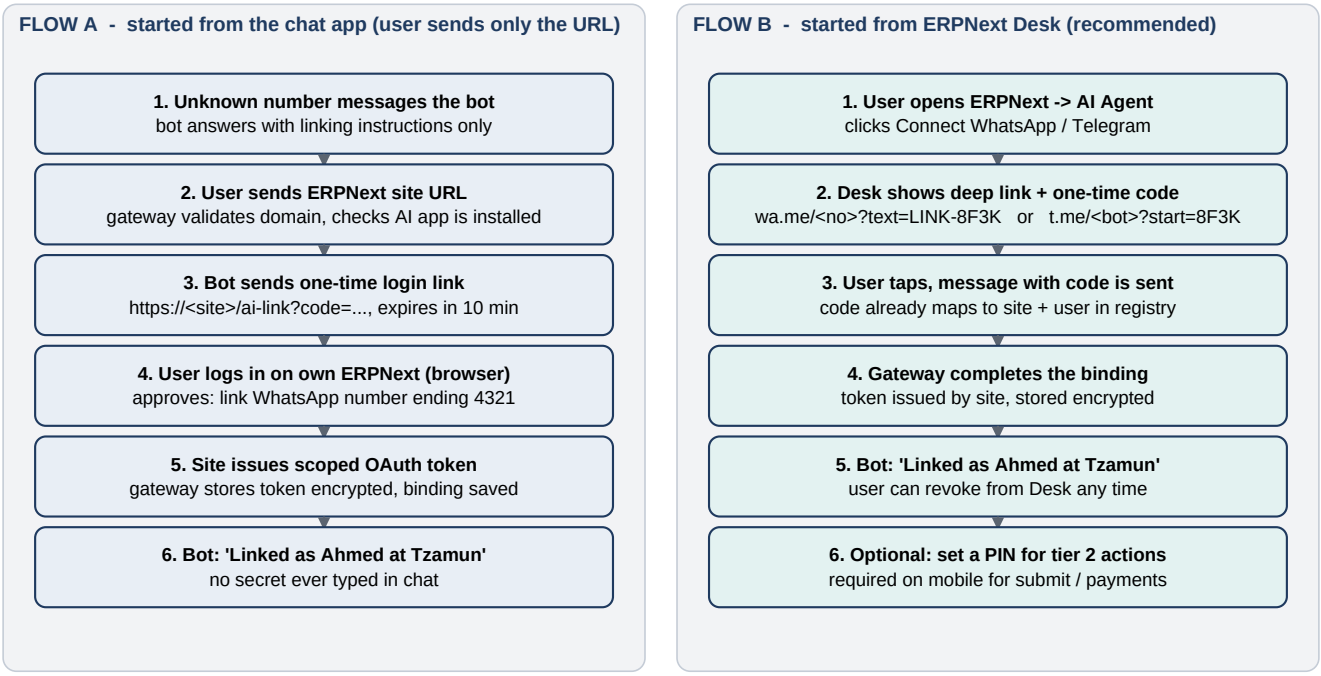


Figure 3. The two linking flows. The user types at most the site URL; identity is proven by logging in to their own ERPNext.

Act-as-user worker

- The worker resolves the binding, calls frappe.set_user for the bound user, runs the agent, and resets the user in a finally block.
- The impersonation is written to the audit record with channel, external id hash and job id.
- This code path is small, isolated in one module, and covered by tests that assert permissions are enforced for the impersonated user.

- Bindings carry an inactivity expiry set by the company, a device label, and a one-active-device-per-user option.

Extra controls

- Site URL allow-list on the shared gateway for enterprise customers, so a phished user cannot be linked to a look-alike server.
- Tier 2 actions on messaging channels require a PIN set during linking, or a re-confirmation link that opens in Desk.
- The gateway registry stores site URL, user, phone hash and token only. No message content is retained beyond the delivery queue.
- All OAuth tokens are scoped to the AI agent endpoints and cannot be used against the general REST API.

8. Layer 4: Agent runtime

The runtime is a thin, typed orchestration loop. It builds the prompt from the system instructions, the channel profile, the user's context and the conversation memory, asks the provider for a completion with the allowed tool schemas, routes each tool request through the policy engine, executes or defers it, feeds results back, and renders the final answer.

Components

Component	Detail
Orchestrator	Pydantic AI agent with typed tools. Maximum step count per turn, per-tool timeouts, and a total token budget per turn. Falls back to a clarifying question when the model loops.
System prompt assembly	Base instructions plus company profile (name, currencies, fiscal year), user profile (roles, default company, language), channel profile (short answers on mobile), and the current Desk context when available. Arabic or English selected from the user's language setting.
Conversation memory	Stored in AI Conversation and AI Message DocTypes keyed by user and channel thread. A rolling window of recent messages plus a compact summary of older turns. Desk threads and WhatsApp threads are separate conversations.
Pending Action store	Tier 2 tool requests are persisted as Pending AI Action records with the exact arguments, a hashed single-use token and an expiry. Execution resumes from the database when the confirmation arrives, minutes or hours later, on any channel. This is deliberately independent of any framework's in-memory approval feature.
Renderer	Streams tokens to Desk over Socket.IO using frappe.publish_realtime. Buffers the full answer for messaging channels. Emits structured cards for proposed documents so Desk can show a preview and WhatsApp can show a summary.
Entity resolution	Before creating documents, resolves names to link values through the search tools and asks the user when more than one plausible match exists.
Guardrails	Tool outputs are capped in size and stripped of HTML. Content from documents, comments and emails is marked as untrusted data in the prompt. The model cannot request tools not in the allow-list for this user and channel.

Turn lifecycle

1. Receive InboundMessage with user, channel, text, attachments and optional Desk context.
2. Load or create the conversation; load memory window and summary.
3. Resolve the allowed tool set for this user and channel from the policy engine.
4. Call the provider with messages and tool schemas. Stream if the channel supports it.
5. For each tool request: policy check, then execute (tier 0 and 1), defer (tier 2) or refuse (tier 3 or not allowed). Record an AI Tool Call.
6. Return tool results to the model until it produces a final answer or the step budget is hit.
7. Render the answer for the channel, attach cards or files, and send.
8. Persist messages, token usage and latency. Update the conversation summary in a background job when the window is full.

9. Layer 5: Policy engine

Frappe permissions decide what the user may do. The policy engine decides what the agent may do on the user's behalf, which is always a subset. It never grants anything Frappe would refuse.

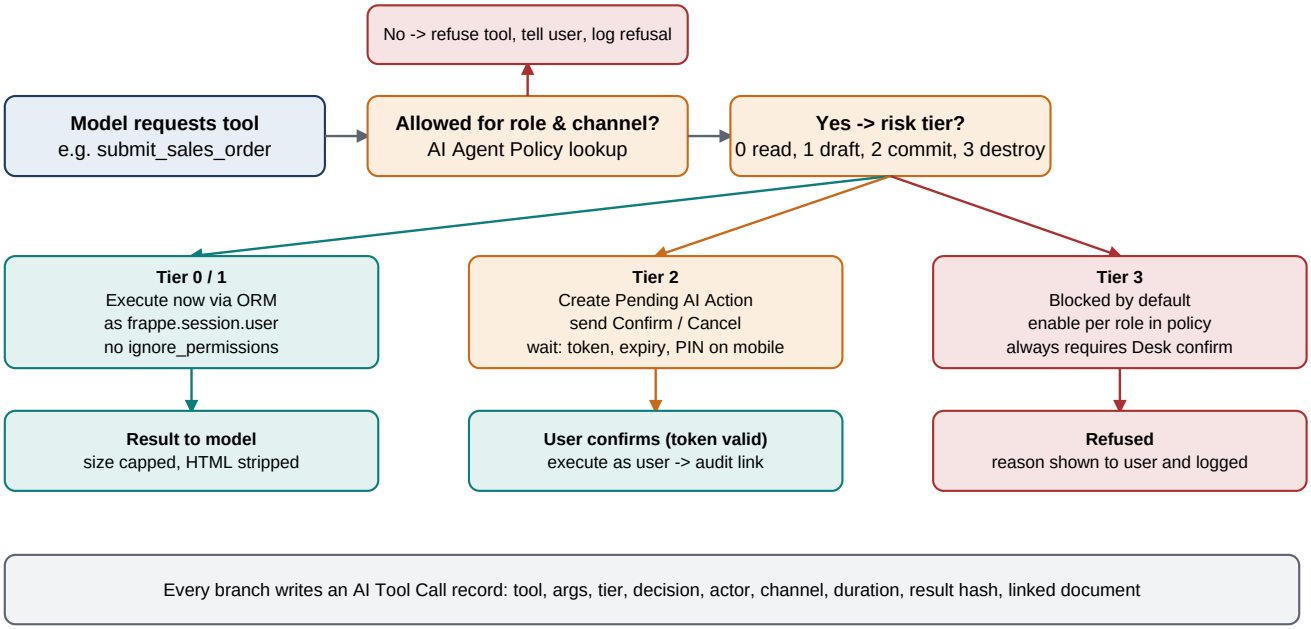


Figure 4. Policy decision for every tool request.

Risk tiers and channel defaults

Tier	Actions	Desk	WhatsApp / Telegram	Self-hosted messenger	External MCP client
0 Read	get, list, search, reports, ledger summaries	Auto-run	Auto-run, with data classification limits (no salary, bank, cost fields unless allowed)	Auto-run	Auto-run
1 Draft	create draft documents, add comments, assign	Auto-run, shown as card with link	Auto-run, summary sent	Auto-run	Auto-run
2 Commit	submit, make payment, stock movement, workflow approve	Confirm button with preview	Buttons plus PIN, or link to Desk	Confirm button	Client approval plus server-side pending action
3 Destroy	cancel, delete, amend, change permissions	Off by default; enable per role; Desk confirm only	Off	Off by default	Off

Policy record dimensions

- **Role x DocType allow-list.** Which DocTypes are exposed to the agent for which roles. Default set ships for Sales, Purchase, Stock, Accounts and HR.
- **Maximum tier per role and channel.** For example, Sales User may reach tier 2 on Desk but only tier 1 on WhatsApp.
- **Blocked fields.** Field names that are removed from tool results for a channel, on top of Frappe's permlevel rules.
- **Budgets.** Tokens per user per day, calls per minute, and a company-wide monthly cap with alerts at 80 percent.
- **Working hours and geography.** Optional restriction of tier 2 actions on mobile to business hours or known countries.

Implementation notes

- Policy is evaluated in Python before the tool runs, and again inside the tool for defence in depth.
- All tools use frappe.get_doc, frappe.get_list, doc.insert, doc.submit and frappe.has_permission. The flag ignore_permissions is banned by a lint rule in the app.
- No raw SQL tool is exposed to the model. Reports run through Frappe's report engine, which applies permissions.
- Refusals are explained to the user in plain language and logged with the reason.

10. Layer 6: Tool registry and MCP server

Tools are plain Python functions with typed arguments and docstrings, registered once. The same registry serves two transports: in-process calls from the embedded agent, and the Model Context Protocol for external clients. Tool schemas for the LLM are generated from the type hints, so there is one source of truth.

Tool design rules

- Few, specific, well-described tools beat one generic "do anything" tool. Version 1 ships 18 tools listed in Appendix A.
- Every tool declares its DocType, risk tier and required Frappe permission type in a decorator, so policy is data, not scattered code.
- Read tools return compact, permission-filtered dictionaries with a size cap and a "more available" flag.
- Write tools accept a validated schema, create drafts by default, and return the document name plus a Desk link.
- Tools are idempotent where possible and carry an idempotency key from the tool call id to survive retries.
- Every tool call writes an AI Tool Call record before and after execution.

Tool declaration example

```
@ai_tool(doctype="Sales Order", tier=1, perm="create")
def create_sales_order_draft(customer: str, items: list[ItemLine], delivery_date: date, company:
str | None = None) -> ToolResult:
    """Create a draft Sales Order for a customer. Does not submit. Returns name and link."""
```

MCP server

Aspect	Decision
Implementation	Official Python MCP SDK, FastMCP style server, wrapping the same registry.
Transport	Streamable HTTP, served by a dedicated process under supervisor behind nginx at /mcp. Stdio transport available for local developer tools.
Authentication	OAuth 2.1 with the Frappe site as the authorisation server. Bearer token maps to a Frappe user; the server sets that user for the request.
Policy	Same policy engine. Tier 2 tools return a pending action id and a confirmation URL; the client shows the user the link or the server sends the approval to the user's bound channel.
Resources and prompts	MCP resources expose read-only views such as company profile and DocType schemas. MCP prompts ship reusable task templates such as month-end summary.
Discovery	Tool list is filtered per user and channel, so an external client only sees what that user may use.

11. Layer 7: Knowledge and RAG

Two kinds of knowledge are handled differently. Structured data such as orders, invoices and stock levels is retrieved with tools because those questions need exact filters and live values. Unstructured text such as knowledge base articles, standard operating procedures, policies, item descriptions and email threads is retrieved with embeddings.

Stage	Detail
Sources	Knowledge Base and Help Article DocTypes, attached files (PDF, DOCX) on selected DocTypes, Item descriptions, Communication and Comment text, optional external folders. Each source is switched on per DocType in AI Settings.
Ingestion	Document event hooks (after_insert, on_update, on_trash) enqueue background jobs. Text is extracted, cleaned, chunked at roughly 400 tokens with overlap, embedded and upserted into the vector store with metadata.
Metadata	Source DocType and name, owner, roles allowed, company, language, updated timestamp. Used for permission filtering and freshness.
Embeddings	Public: OpenAI text-embedding-3-large. Private: BGE-M3 served by Text Embeddings Inference or Ollama. BGE-M3 is chosen for strong Arabic and English performance and long inputs.
Vector store	Qdrant, self-hosted in one container. Payload filters implement permission-aware retrieval. Abstracted behind a small interface so MariaDB Vector can replace it later.
Retrieval	Hybrid: dense vectors plus keyword match on chunk text, merged with reciprocal rank fusion. Top results are re-checked with frappe.has_permission on the source document before being shown to the model.
Citations	Every retrieved chunk is cited back to its source document with a Desk link so users can verify answers.
Freshness	Chunks are versioned by document modified timestamp; stale chunks are deleted on update. A nightly job reconciles the index against the database.

What RAG is not used for

- Numeric questions such as outstanding receivables. Those go through report tools for exact answers.
- Anything the user cannot read. Permission filtering is applied at query time and re-verified per result.
- Cross-company leakage. Company is a mandatory filter on every query.

12. Layer 8: LLM provider layer

One interface with several drivers. Switching from Azure OpenAI to vLLM is a change in AI Settings and a restart of the workers. Every supported combination is shipped as a model profile with tested defaults and evaluation results, so a customer can choose "private, Qwen3 32B on vLLM" and know what to expect.

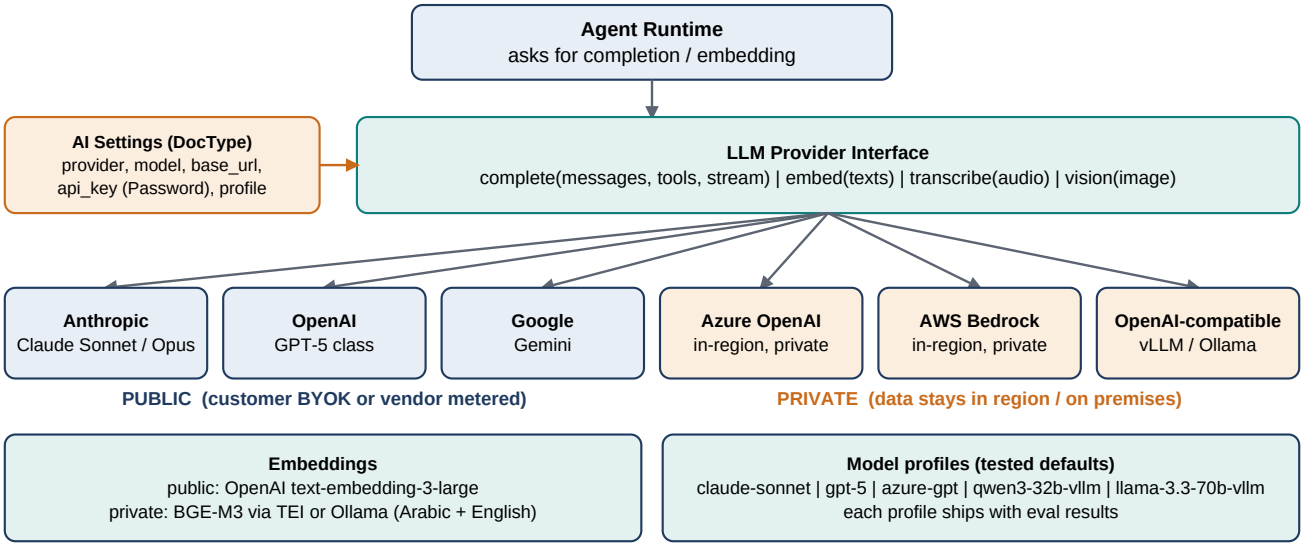


Figure 5. Provider abstraction with public and private drivers selected by configuration.

Model profiles

Profile	Tier	Chat model	Embeddings	Transcription / vision	Notes
claude-sonnet	Public	Claude Sonnet, Opus for complex turns	OpenAI text-embedding-3-large or Voyage	Provider vision; OpenAI Whisper API	Best tool-use reliability; default for startups
gpt-5	Public	GPT-5 class	text-embedding-3-large	Provider vision; Whisper API	Alternative default
gemini	Public	Gemini Pro	Gemini embeddings	Provider vision and audio	Strong multimodal
azure-openai	Private cloud	GPT-5 class on Azure, in-region	Azure embeddings	Azure Whisper and vision	Data stays in customer tenant and region; check regional availability
bedrock-claude	Private cloud	Claude on Bedrock, in-region	Titan or Cohere embeddings	Bedrock vision; Transcribe	Same as above on AWS
qwen3-32b-vllm	On premises	Qwen3 32B on vLLM	BGE-M3 on TEI	Whisper large-v3 local; Qwen VL	One 24 GB GPU at 4-bit; good tool calling
llama-3.3-70b-vllm	On premises	Llama 3.3 70B on vLLM	BGE-M3 on TEI	Whisper local; Llama vision	Two 80 GB GPUs; strongest local option
ollama-dev	Development	Any Ollama model	BGE-M3 on Ollama	Whisper local	Laptops and pilots only, not production

Interface contract

- complete(messages, tools, stream, max_tokens, temperature) returning text deltas and structured tool calls in one normalised shape.
- embed(texts) returning vectors with the model name and dimension recorded so an index cannot mix models.
- transcribe(audio, language_hint) and vision(image, prompt) for media, with local fallbacks in the private tier.

- Retries with backoff, timeout per call, and circuit breaker per provider. Usage is written to AI Usage Log after every call.
- Optional LiteLLM proxy driver for customers who already run a central LLM gateway with their own keys and logging.

13. Layer 9: Audit, observability and evaluation

Audit trail

The audit answers four questions for any AI-touched document: who asked, what the model proposed, who approved, and what changed. It is built on DocTypes so it inherits Frappe's own permissions, reports and retention tools.

Record	Written when	Key fields
AI Conversation	A new thread starts	user, channel, thread id, Desk context, start and last active time, status
AI Message	Each user or assistant message	role, content, attachments, model, tokens in and out, latency
AI Tool Call	Before and after every tool request	tool, arguments, tier, policy decision, actor, channel, duration, result hash, linked DocType and name, error
Pending AI Action	A tier 2 tool is deferred	tool call, token hash, expiry, status, confirmed by, confirmation channel, PIN verified
Impersonation entry	A worker acts as a bound user	user, channel, external id hash, job id, start and end
Document link	A tool creates or changes a document	A Comment on the document's timeline and a link field back to the AI Tool Call
AI Usage Log	Every provider call	user, company, day, provider, model, tokens, estimated cost

Observability

- Structured logs with conversation id and tool call id on every line, shipped to the customer's existing log stack.
- Metrics: turns per minute, tool error rate, approval rate, median latency per provider, tokens per company per day.
- Dashboards inside Desk built on the audit DocTypes: usage by user, top tools, refusals, pending approvals ageing.
- Alerts on budget thresholds, provider outages and unusual patterns such as many refusals from one binding.

Evaluation suite

Because the product promises a public and private toggle, every model profile must be certified against the same tests. The suite runs in CI against a seeded demo site.

- About 150 prompts in Arabic and English with the expected tool calls and arguments, covering each tool and common ambiguities.
- Safety cases: prompt injection embedded in a customer note, requests beyond the user's permissions, tier 3 attempts on mobile.
- Scoring: exact tool match, argument accuracy, refusal correctness, answer faithfulness to tool results, latency and cost.
- A profile ships only when it clears the threshold; results are printed on the profile so customers can compare.

14. Layer 10: Platform, deployment and sizing

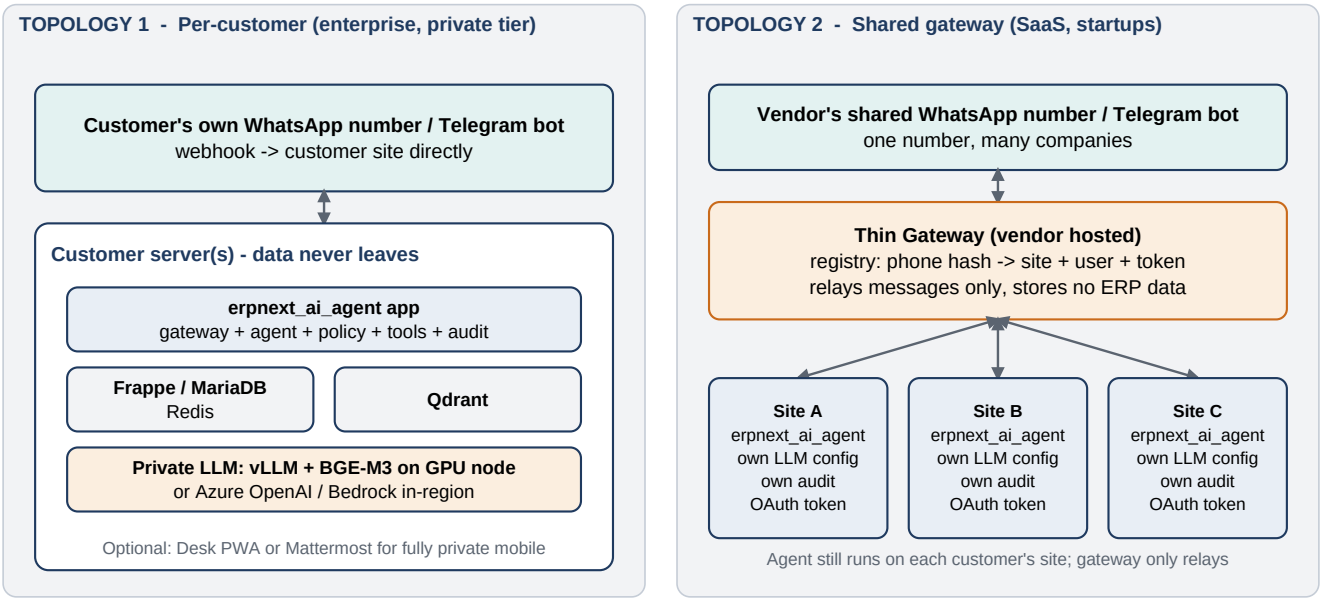


Figure 6. Deployment topologies. Both share one codebase; the agent always runs on the customer's site.

Topology choice

	Per-customer	Shared gateway
Who	Enterprises, regulated customers, private tier	Startups and small businesses on the vendor's hosted plan
WhatsApp number	Customer's own, verified under their business	Vendor's number, one for all tenants
Where the agent runs	Customer site	Customer site; gateway only relays
Data path	Phone to Meta to customer site	Phone to Meta to vendor gateway to customer site
What the vendor holds	Nothing	Phone hash, site URL, user, encrypted token
Onboarding effort	Higher: Meta verification per customer	Low: link from Desk in one minute

Processes on a Frappe bench

- Web workers handle Desk chat requests and webhooks; they only enqueue.
- A dedicated RQ queue named ai with its own workers runs agent turns, ingestion and summaries, so ERP background jobs are not starved.
- The MCP server runs as a separate supervisor program behind nginx at /mcp.
- Qdrant runs as a container or system service next to Redis. TEI and vLLM run on the GPU node when the private tier is used.
- Socket.IO streams tokens to Desk through Frappe's existing realtime server.

Private model sizing

Workload	Model	Minimum hardware	Concurrency guide
Pilot, up to 20 users	Qwen3 32B, 4-bit on vLLM	One 24 GB GPU (RTX 4090 or L4 class), 64 GB RAM	Two to four concurrent turns
Production, up to 200 users	Qwen3 32B, 8-bit or Llama 3.3 70B, 4-bit	Two 48 GB GPUs or one 80 GB GPU	Eight to twelve concurrent turns
Production, strongest quality	Llama 3.3 70B, 16-bit	Two 80 GB GPUs (A100 or H100)	Twelve plus with continuous batching
Embeddings and transcription	BGE-M3, Whisper large-v3	Shares the GPU or runs on CPU for small sites	Batch jobs, not latency critical

Figures are planning guides. Final sizing is confirmed in the phase 2 load test against the eval suite.

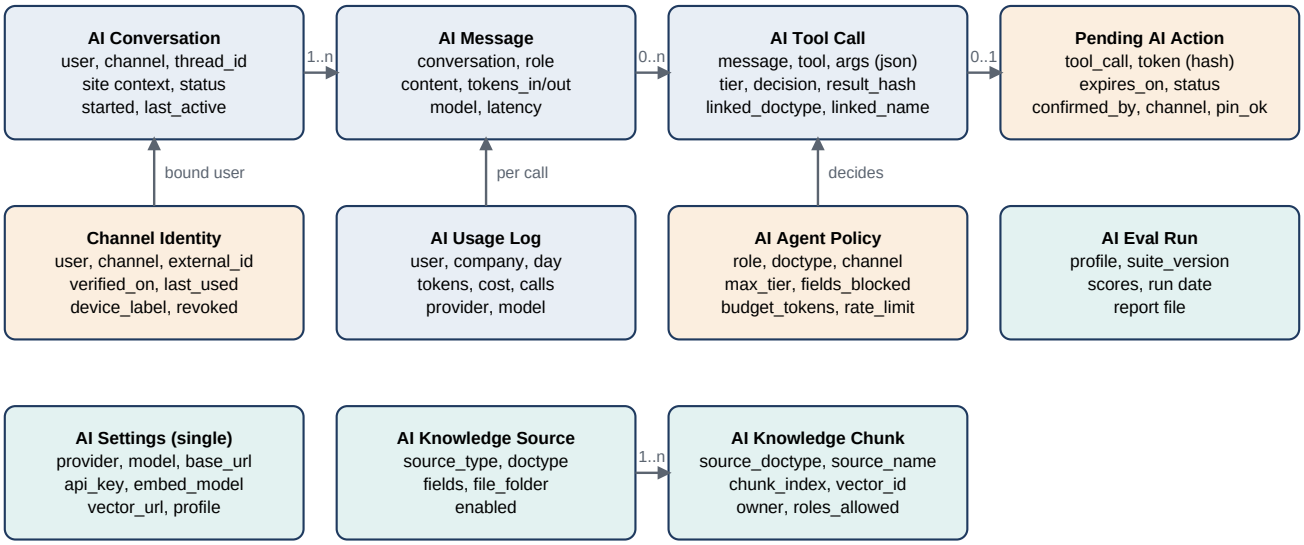
15. Security and threat model

Threat	Mitigation	Layer
Prompt injection through ERPNext data (a customer note says "delete all invoices")	Tool outputs marked as untrusted data; write actions gated by tier and confirmation; tool allow-list; no generic SQL tool; injection cases in the eval suite	4, 5, 9
Agent exceeds user permissions	All tools via the Frappe ORM as the real user; ignore_permissions banned by lint; policy is a subset of Frappe permissions; permission re-check per RAG result	5, 6, 7
Credential theft	No ERP credential ever leaves the server; LLM keys in Password fields decrypted only server-side; OAuth tokens scoped, encrypted, expiring, revocable	3, 8
Lost or shared phone	Binding revocation from Desk; inactivity expiry; PIN for tier 2 on mobile; one active device option; data classification limits on channels	2, 3, 5
Unknown senders and cost abuse	Unbound senders never reach the LLM; per-user and company budgets; rate limits per binding	2, 5
Phishing to a look-alike site	Site URL allow-list on the shared gateway; linking link shown with full domain; Desk-initiated flow preferred	3
Data leaving the region	Private tier with in-region cloud or on-premises models; policy can disable WhatsApp and Telegram; self-hosted messenger option	1, 8
Cross-tenant leakage on the shared gateway	Gateway stores no ERP data; per-tenant tokens; company filter mandatory in retrieval; separate conversations per site	2, 7
Replayed or forged webhooks	Signature verification; message id dedupe; single-use approval tokens with expiry	2
Model hallucination on numbers	Numeric questions answered only from tool results; citations for text answers; faithfulness scoring in evals	4, 9
Insider misuse of the agent	Full audit with actor, channel and approver; dashboards on refusals and unusual activity; audit DocTypes protected by role	9
Supply chain and dependencies	Pinned versions, vulnerability scanning in CI, minimal dependency set, no dynamic code execution tools	10

Compliance notes

- Saudi PDPL: purpose limitation is met by the tool allow-list; data residency by the private tier; subject access by the audit DocTypes.
- Retention: conversation and audit records follow a configurable retention policy with export before purge.
- Third-party processors: when public LLMs are used, the customer's contract with the provider must cover it; the app shows which provider is active on every conversation.

16. Data model



blue: conversation and audit orange: control green: configuration and knowledge

Figure 7. DocTypes introduced by the app and their relationships.

DocType	Purpose	Notable fields
AI Settings (Single)	Provider and channel configuration	provider, model, base_url, api_key (Password), embedding provider and model, vector store URL, active profile, channel toggles, WhatsApp and Telegram credentials (Password), retention days
AI Agent Policy	What the agent may do per role, DocType and channel	role, doctype, channel, max_tier, blocked_fields, daily_token_budget, calls_per_minute, business_hours_only
Channel Identity	Binding of a platform identity to a Frappe user	user, channel, external_id (hashed and encrypted), verified_on, last_used, device_label, pin_hash, revoked
AI Conversation	A thread of messages	user, channel, thread_id, context (json), status, summary
AI Message	One message in a thread	conversation, role, content, attachments, model, tokens_in, tokens_out, latency_ms
AI Tool Call	One tool request and its outcome	message, tool, arguments (json), tier, decision, actor, channel, duration_ms, result_hash, linked_doctype, linked_name, error
Pending AI Action	A deferred tier 2 action	tool_call, token_hash, expires_on, status, confirmed_by, confirmation_channel, pin_verified
AI Knowledge Source	Which DocTypes and folders are indexed	source_type, doctype, fields, file_folder, enabled
AI Knowledge Chunk	Index bookkeeping per chunk	source_doctype, source_name, chunk_index, vector_id, owner, roles_allowed, company, language, modified
AI Usage Log	Token and cost accounting	user, company, date, provider, model, tokens_in, tokens_out, cost_estimate, calls
AI Eval Run	Certification results per profile	profile, suite_version, score fields, run date, report file

17. Technology stack

Layer	Choice	Why	Alternatives considered
Platform	Frappe v15 and later, Python 3.11 and later, one custom app erpnext_ai_agent	Native permissions, hooks, background jobs, realtime, OAuth provider	Standalone service next to Frappe: rejected, loses permissions and session
Agent runtime	Pydantic AI	Typed tools, provider agnostic, MCP client support, small surface, easy to test	LangGraph for complex multi-step graphs later; CrewAI rejected for enterprise use
Provider layer	Own thin adapter over native Anthropic, OpenAI and Gemini SDKs plus an OpenAI-compatible driver; optional LiteLLM proxy driver	Covers vLLM, Ollama, LM Studio, Azure, Bedrock; full control of retries and logging	LiteLLM as the only layer: rejected, too much surface for a product dependency
MCP	Official Python MCP SDK, FastMCP style, Streamable HTTP, OAuth 2.1 via Frappe	Standard, maintained, one registry for both transports	Custom protocol: rejected
Vector store	Qdrant self-hosted, behind a small interface	Payload filters for permission-aware retrieval, single container, good performance	MariaDB Vector 11.7 and later once common on Frappe installs; pgvector not applicable
Embeddings	OpenAI text-embedding-3-large (public); BGE-M3 on Text Embeddings Inference (private)	BGE-M3 handles Arabic and English; TEI is fast and simple	Nomic embed, Cohere multilingual
Private serving	vLLM in production, Ollama for development	Continuous batching, OpenAI-compatible API, quantisation support	LM Studio for desktops only
Transcription and vision	Provider APIs publicly; Whisper large-v3 and a vision-capable open model privately	Voice notes and invoice photos are core mobile features	
Desk UI	Frappe page and floating widget built with Vue through Frappe's build pipeline; PWA manifest	No new frontend stack; works on phones	Separate React app: rejected
Messaging	WhatsApp Cloud API directly or via a BSP; Telegram Bot API; Mattermost and Rocket.Chat integrations	Reference plumbing from community apps such as frappe_whatsapp	
Queues and realtime	Redis, RQ workers on a dedicated ai queue, Socket.IO via Frappe	Already present on every bench	
Testing and CI	pytest with a seeded demo site, eval suite runner, ruff lint including the ignore_permissions rule, dependency scanning	Certifies each model profile	

18. Roadmap and milestones

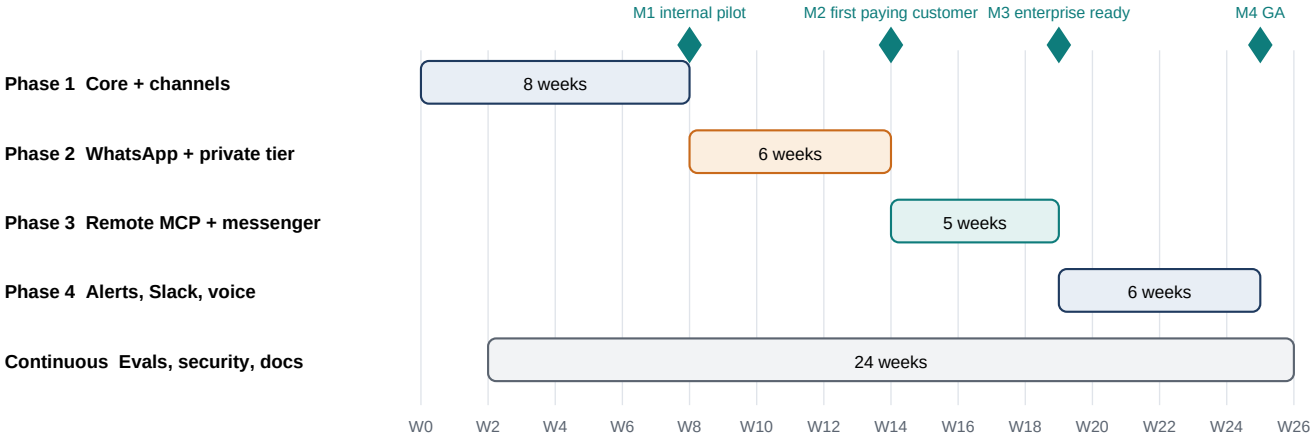


Figure 8. Four delivery phases over about 25 weeks with continuous evaluation and security work.

Phase 1: Core and channel foundation (weeks 1 to 8)

- App skeleton, AI Settings, AI Agent Policy, audit DocTypes, Channel Identity, Pending AI Action.
- Provider layer with Anthropic and OpenAI drivers and the OpenAI-compatible driver; two model profiles certified.
- Tool registry with the 18 initial tools; policy engine with tiers; lint rule against ignore_permissions.
- Desk chat widget with streaming, context awareness, preview cards and confirmations; PWA manifest.
- Telegram connector with Desk-initiated linking, inline keyboard approvals, voice note transcription.
- Eval suite v1 with about 100 prompts; CI pipeline; seeded demo site.
- Milestone M1: internal pilot on the vendor's own ERPNext.

Phase 2: WhatsApp and private tier (weeks 8 to 14)

- WhatsApp Cloud API connector: verification guide, templates, reply buttons, PIN flow, image intake with vision extraction.
- Private tier: vLLM and TEI deployment guides, BGE-M3, Qdrant, Whisper local; profiles qwen3-32b-vllm and llama-3.3-70b-vllm certified.
- RAG ingestion for Knowledge Base, Help Articles and attached files with permission-filtered retrieval and citations.
- Load test and sizing confirmation; budgets and rate limits; usage dashboards.
- Milestone M2: first paying customer on the startup tier.

Phase 3: Remote MCP and private messenger (weeks 14 to 19)

- MCP server over Streamable HTTP with OAuth 2.1; verified with Claude, ChatGPT and an IDE client.
- Shared gateway service with tenant registry, URL allow-list and Flow A linking.
- Mattermost and Rocket.Chat connector for the private tier; policy switches to disable public messengers.
- External security review and penetration test; compliance documentation pack.
- Milestone M3: enterprise ready.

Phase 4: Proactive alerts, Slack and voice (weeks 19 to 25)

- Scheduled digests and threshold alerts over approved channels using WhatsApp templates.
- Slack connector; voice calls through realtime speech APIs publicly and local speech privately.
- Admin console improvements, retention tooling, marketplace listing and documentation.
- Milestone M4: general availability.

19. Team and effort

Role	Allocation	Owns
Frappe backend lead	Full time	App structure, DocTypes, permissions, tools, policy engine, act-as-user worker
AI engineer	Full time	Agent runtime, provider layer, RAG, model profiles, eval suite
Integration engineer	Full time	Channel gateway, WhatsApp and Telegram connectors, MCP server, shared gateway
Frontend engineer	Full time	Desk widget, PWA, preview cards, admin dashboards
DevOps	Half time	vLLM and TEI deployment, Qdrant, CI, load tests, private tier guides
QA and security	Half time	Test plans, eval runs, penetration test coordination
Product manager	Half time	Tool catalogue priorities, customer pilots, packaging, documentation

Effort summary

Phase	Duration	Engineer weeks	Main risk
1 Core and channel foundation	8 weeks	About 40	Tool-calling quality and policy edge cases
2 WhatsApp and private tier	6 weeks	About 30	Meta verification delays; GPU procurement
3 Remote MCP and private messenger	5 weeks	About 25	OAuth interoperability with external clients
4 Proactive alerts, Slack, voice	6 weeks	About 30	Template approvals; voice latency

20. Commercial packaging

Tier	Target	LLM	Channels	Hosting	Metering
Startup	Small companies	Public, bring your own key	Desk, Telegram, WhatsApp via shared gateway	Vendor hosted or self-hosted	Per user per month; customer pays provider directly
Business	Mid-size companies	Public, vendor-managed keys	Desk, Telegram, WhatsApp via shared gateway or own number	Vendor hosted or self-hosted	Per user plus token bundles
Enterprise	Regulated and large companies	Private: in-region cloud or on premises	Desk PWA, own WhatsApp and Telegram, Mattermost or Rocket.Chat, MCP gateway	Self-hosted, air-gap capable	Annual licence plus deployment services

One codebase serves all tiers. Differences are settings, profiles and which connectors are enabled by the licence key.

21. Risks and mitigations

Risk	Likelihood	Impact	Mitigation
Local models call tools unreliably	Medium	High	Certify profiles with the eval suite; recommend 30B or larger; constrained decoding for tool arguments in vLLM
WhatsApp business verification delays customers	High	Medium	Telegram and Desk PWA available from day one; BSP partner; shared gateway for startups
Users type secrets into chat anyway	Medium	High	Bot detects key-like strings and refuses, deletes where the platform allows, and sends the linking guide
Prompt injection leads to unwanted writes	Medium	High	Tier gating, confirmations, allow-list, injection tests in CI
Provider price or policy changes	Medium	Medium	Provider layer makes switching a configuration change; profiles for at least two public and two private options
GPU cost surprises for enterprises	Medium	Medium	Sizing table, load test in phase 2, in-region cloud option as a middle path
Frappe upstream adds overlapping AI features	Medium	Medium	Keep the tool registry and policy independent so they can plug into upstream chat surfaces; track Frappe releases
Meta or Telegram policy rejects business bots	Low	High	Self-hosted messenger path is fully supported
Audit volume grows large	High	Low	Retention policy, archival job, summary tables for dashboards

22. Open decisions

Decision	Options	Recommendation	Needed by
WhatsApp access route	Meta Cloud API directly, or a Business Solution Provider	BSP for the shared gateway; direct for enterprise customers who already have a WABA	Phase 2 start
Default public model	Claude Sonnet, GPT-5 class, Gemini	Claude Sonnet for tool reliability; let customers switch	Phase 1 week 2
Vector store long term	Qdrant, MariaDB Vector	Qdrant now; revisit when most installs run MariaDB 11.7 or later	Phase 2
PIN versus Desk link for mobile tier 2	PIN in chat, link to Desk, both	Both, chosen per company in policy	Phase 1 week 6
Shared gateway hosting region	Saudi Arabia, UAE, EU	Saudi Arabia for PDPL alignment	Phase 3
Licence enforcement	Licence key DocType, hosted licence server	Signed licence key with offline validation for air-gapped sites	Phase 3

Appendix A: Initial tool catalogue

Tool	DocType	Tier	Purpose
search_entities	Customer, Supplier, Item, Employee, Project	0	Resolve a name to link values; returns candidates for disambiguation
get_document	Any allowed	0	Fetch one document with permission-filtered fields
list_documents	Any allowed	0	Filtered, paginated list with size cap
get_doctype_schema	Any allowed	0	Field names, types and options so the model builds valid documents
run_report	Report	0	Run a query or script report with filters; returns rows with cap
get_stock_balance	Bin, Stock Ledger	0	Stock for item and warehouse, with projected quantity
get_receivables_payables	GL, Sales and Purchase Invoice	0	Outstanding by party with ageing
summarise_ledger	GL Entry	0	Ledger summary for account, party and period
get_employee_info	Employee, Leave, Attendance	0	Leave balance, attendance, basic profile within permissions
create_sales_order_draft	Sales Order	1	Draft from customer, items, dates
create_quotation_draft	Quotation	1	Draft quotation
create_purchase_order_draft	Purchase Order	1	Draft from supplier and items
create_invoice_draft	Sales Invoice, Purchase Invoice	1	Draft invoice, optionally from an order or an uploaded image
create_leave_application	Leave Application	1	Draft leave request for the user or a report
create_expense_claim_draft	Expense Claim	1	Draft from a receipt photo or text
add_comment_or_assign	Any allowed	1	Comment on or assign a document
submit_document	Any allowed	2	Submit a draft after confirmation
create_payment_entry	Payment Entry	2	Record a payment against invoices after confirmation
make_stock_entry	Stock Entry	2	Transfer or issue stock after confirmation
approve_workflow_action	Workflow	2	Apply a workflow action after confirmation
search_knowledge	Knowledge sources	0	Permission-filtered retrieval with citations

Tier 3 tools such as cancel_document and delete_document are implemented but disabled by default and only enabled per role by an administrator.

Appendix B: AI Settings reference

Section	Field	Type	Notes
Provider	active_profile	Select	One of the shipped model profiles
Provider	chat_provider, chat_model, chat_base_url	Data	Overrides for custom endpoints
Provider	chat_api_key	Password	Encrypted; decrypted only server-side
Provider	embedding_provider, embedding_model, embedding_base_url, embedding_api_key	Data / Password	Index records the model name
Provider	transcription_provider, vision_provider	Select	Local options in the private tier
Provider	litellm_proxy_url	Data	Optional gateway
Knowledge	vector_store_url, vector_store_api_key	Data / Password	Qdrant
Knowledge	indexed sources	Table	AI Knowledge Source rows
Channels	enable_desk, enable_telegram, enable_whatsapp, enable_mattermost, enable_mcp	Check	Policy can further restrict per role
Channels	telegram_bot_token, whatsapp_phone_id, whatsapp_access_token, whatsapp_app_secret, whatsapp_verify_token	Password	Per customer
Channels	shared_gateway_url, shared_gateway_site_key	Data / Password	Only for the startup and business tiers
Security	mobile_tier2_mode	Select	pin, desk_link, both
Security	binding_inactivity_days, one_device_per_user, business_hours_only	Int / Check	
Security	blocked_field_patterns	Small Text	Global list added to policy blocks
Limits	default_daily_tokens_per_user, company_monthly_token_cap, calls_per_minute	Int	Alerts at 80 percent
Retention	conversation_retention_days, audit_retention_days	Int	Export before purge

Appendix C: API endpoints

Endpoint	Auth	Purpose
/api/method/erpnext_ai_agent.api.chat.send	Frappe session	Desk widget sends a message; response streams over Socket.IO
/api/method/erpnext_ai_agent.api.chat.confirm	Frappe session	Confirm or cancel a pending action from Desk
/api/method/erpnext_ai_agent.api.link.start	Frappe session	Create a one-time linking code and deep link
/api/method/erpnext_ai_agent.api.link.complete	Frappe session, via browser link	Complete Flow A after login
/api/method/erpnext_ai_agent.webhooks.telegram	Secret token header, guest	Telegram updates
/api/method/erpnext_ai_agent.webhooks.whatsapp	Meta signature, guest	WhatsApp messages and status callbacks
/api/method/erpnext_ai_agent.api.gateway.inbound	Gateway site key plus user OAuth token	Shared gateway relays a message to the site
/mcp	OAuth 2.1 bearer	MCP Streamable HTTP endpoint
/api/method/erpnext_ai_agent.api.admin.usage	Frappe session, System Manager	Usage and audit dashboards data

Appendix D: Glossary

Term	Meaning
MCP	Model Context Protocol, an open standard for exposing tools, resources and prompts to AI clients
RAG	Retrieval augmented generation: fetching relevant text and giving it to the model with the question
BYOK	Bring your own key: the customer supplies their own LLM provider key
Tier 0 to 3	Risk levels for agent actions: read, draft, commit, destroy
Pending AI Action	A stored, unexecuted tier 2 tool request waiting for user confirmation
Channel Identity	A verified link between a messaging platform account and a Frappe user
Shared gateway	A vendor-hosted relay that lets one WhatsApp number serve many customer sites
vLLM	A high-throughput server for running open models on GPUs with an OpenAI-compatible API
TEI	Text Embeddings Inference, a server for embedding models
BGE-M3	A multilingual embedding model with strong Arabic and English performance
BSP	Business Solution Provider, a Meta partner that provides WhatsApp Business API access
PDPL	Saudi Personal Data Protection Law
PWA	Progressive web app: a website installable on a phone home screen