AI Accounting Agent

AI-powered email triage, drafting and follow-up automation for accounting firms

Serhat Keskin

May 2026

Contents

1 The Problem

Mid-sized accounting and bookkeeping firms manage thousands of client emails per month: missing invoice data, document requests, follow-ups on unpaid balances, regulatory questions. Most of this traffic is routine, repetitive, and language-specific — yet every reply still has to be drafted by a licensed accountant who is, by definition, the most expensive resource in the firm.

> The bottleneck is not knowledge — it is the time it takes a senior accountant to read a thread, locate the missing piece, write a polite reply in the client's language, and remember to follow up next week.

The brief was clear:

- Cut hours spent triaging and replying to client email.
- Never let an outbound message leave the firm without human approval.
- Keep the firm's tone, language coverage (EN / FR / NL) and compliance posture intact.
- Run on the firm's own infrastructure; no client data should leak to a third-party SaaS dashboard.

2 The Solution

AI Accounting Agent is a self-hosted, multi-tenant platform that sits between the firm's mailbox and its accountants. It connects directly to the firm's IMAP/SMTP inbox, analyses every incoming email with Google's Gemini 2.5 Flash model, classifies the request, drafts a context-aware reply, and routes it into a human-in-the-loop approval queue.

> Nothing the agent writes is ever sent automatically. Every outbound message passes through an explicit *Approve* / *Reject* / *Regenerate* step performed by a firm member.

The product covers the full lifecycle of an accounting-firm conversation:

- **Mailbox sync** — IMAP poller pulls new threads, stores them with attachments, runs analysis in a background queue.
- **AI analysis** — detects intent, urgency, missing information (e.g. tax ID, invoice number, KVK / VAT data) and suggests workflow tags.
- **Draft generation** — multilingual HTML replies that match the firm's configured tone and language.
- **Approval workflow** — a generic, UUID-keyed approval pipeline; reviewers can approve, reject, dismiss or regenerate with a hint.
- **Document classification** — uploaded files are auto-tagged (supplier invoice, purchase invoice, identity, other) with confidence scoring.
- **Reminders & follow-ups** — scheduled nudges for unpaid invoices, awaiting documents and unanswered client questions.
- **Usage & cost tracking** — token-level visibility on every AI call, per module, per firm, per day.

3 Architecture

The system is a three-tier stack deployed via Docker Compose behind an Nginx reverse proxy with locally trusted TLS. Eight containers handle isolation, persistence, async work and observability.

3.1 High-level flow

1. Inbound email arrives at the firm's mailbox → `sync_firm_mailbox` Celery task fetches via IMAP.
2. Email + attachments persisted in Postgres → `analyze_and_draft.delay()` enqueued.
3. Worker calls `EmailAIService` (Gemini 2.5 Flash) → analysis, tag assignment, draft generation.
4. Draft + analysis create an `ApprovalRequest` (when `firm.require_approval_emails` is on).
5. Reviewer in the React UI hits `POST /approvals/<uuid>/action/` with `approve / reject / dismiss`.
6. On approve, `ApprovalService._execute_approved_action()` dispatches `send_draft_reply` which sends via SMTP.

3.2 Container topology

Container	Image / Process	Role
`aa-nginx`	nginx 1.29	TLS termination, reverse proxy, SPA hosting
`aa-frontend`	node + Vite	React 19 dev / build server
`aa-backend`	Django 5.2 + Gunicorn	REST API, auth, middleware
`aa-db`	Postgres 16	Multi-tenant relational store
`aa-redis`	Redis 7	Celery broker + cache
`aa-celery-worker`	Celery (gevent, 100c)	Async pipeline workers
`aa-celery-beat`	Celery beat	Scheduled syncs and reminders
`aa-flower`	Flower	Worker monitoring at :5555

3.3 Multi-tenant isolation

Every authenticated request carries a `request.firm` injected by `firms/middleware.py`. Views enforce `IsAuthenticated, IsFirmMember` and filter querysets by `firm=request.firm`; superusers can switch firms via the Firm Admin page. This guarantees that two accounting firms hosted on the same instance cannot see each other's data — a non-negotiable requirement for GDPR-scoped client information.

4 Walkthrough

4.1 Sign-in

The login flow uses HTTP-only refresh cookies (180-day rotation) and a 1-day access token. A silent refresh queue in `src/api/client.ts` retries failed requests after a 401, so reviewers never see a flicker mid-session.

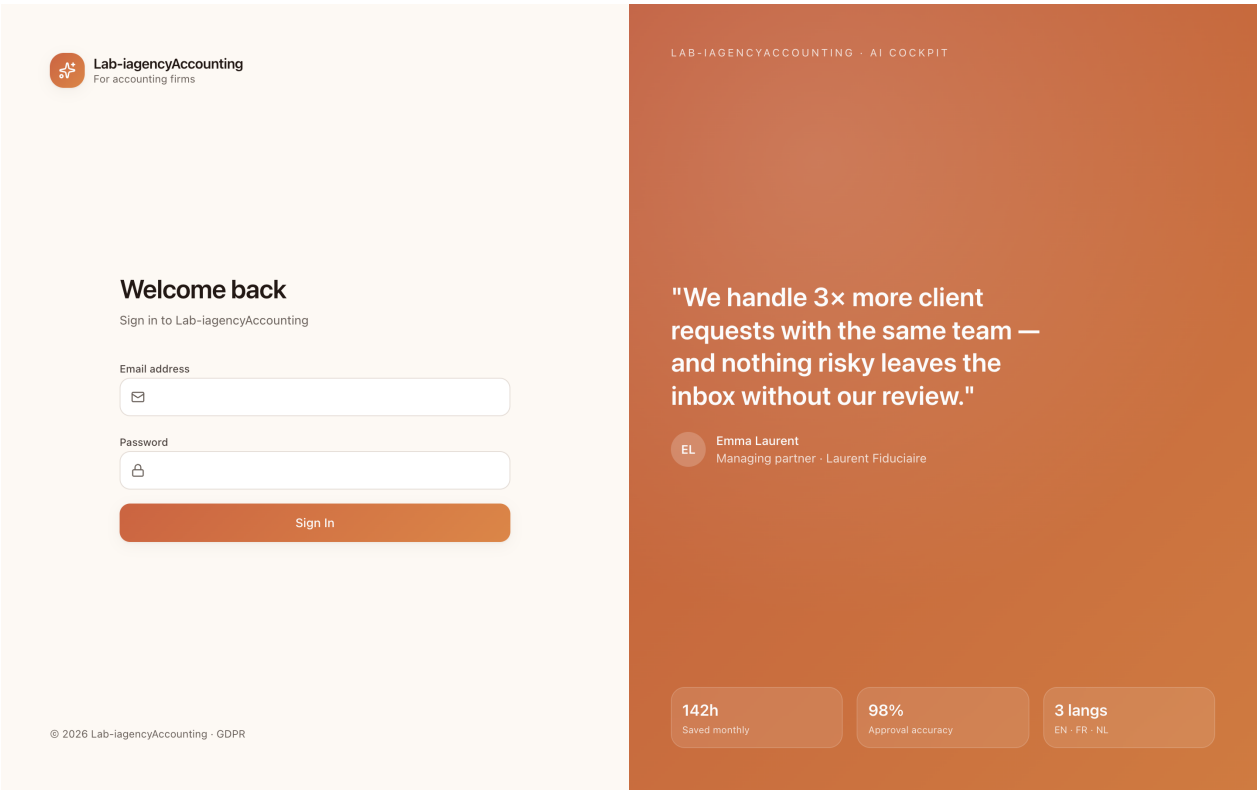

Figure 1 Login screen with cookie-based JWT auth, Google OAuth, and a marketing panel that doubles as a social proof testimonial.

4.2 Dashboard

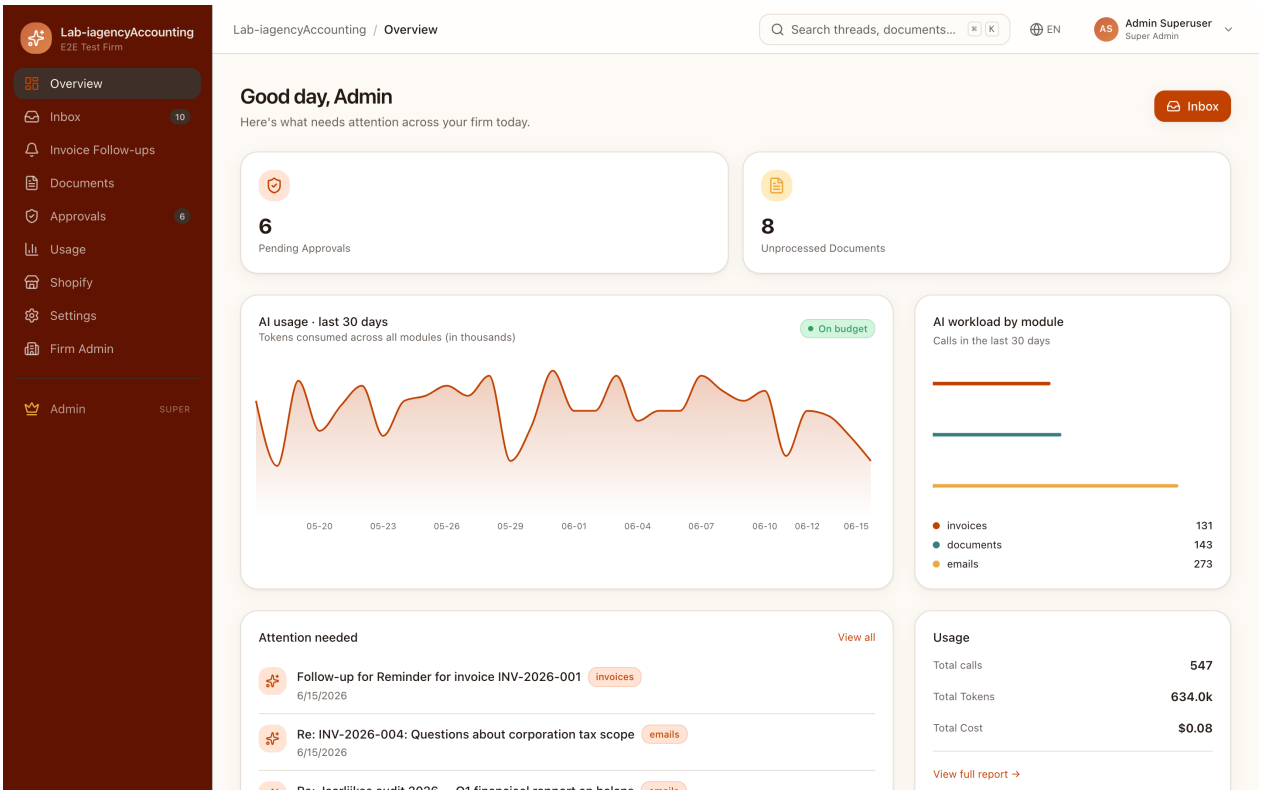

Figure 2 Overview KPIs: pending approvals, unprocessed documents, AI token consumption over the last 30 days, workload split by module, and the attention queue of threads still missing data.

The dashboard is the firm's daily landing pad. It surfaces three signals: *what needs human judgement now* (pending approvals), *what is at risk of being missed* (attention queue), and *what is the cost trajectory* (token + cost spark-lines). All cards link directly into the relevant view.

4.3 Inbox

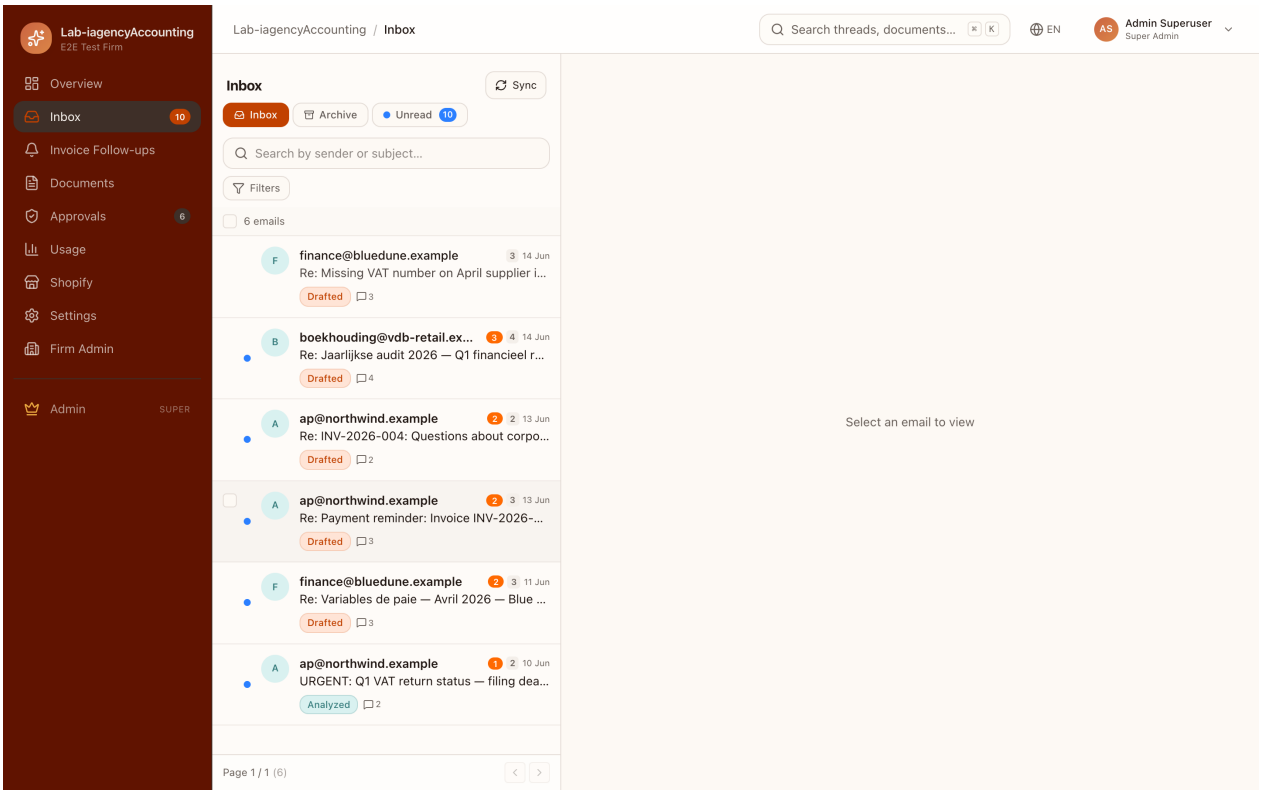

Figure 3 Triaged inbox. Each thread carries colour-coded workflow tags (e.g. *drafted, missing-invoice-data, urgent*) assigned by the AI at sync time.

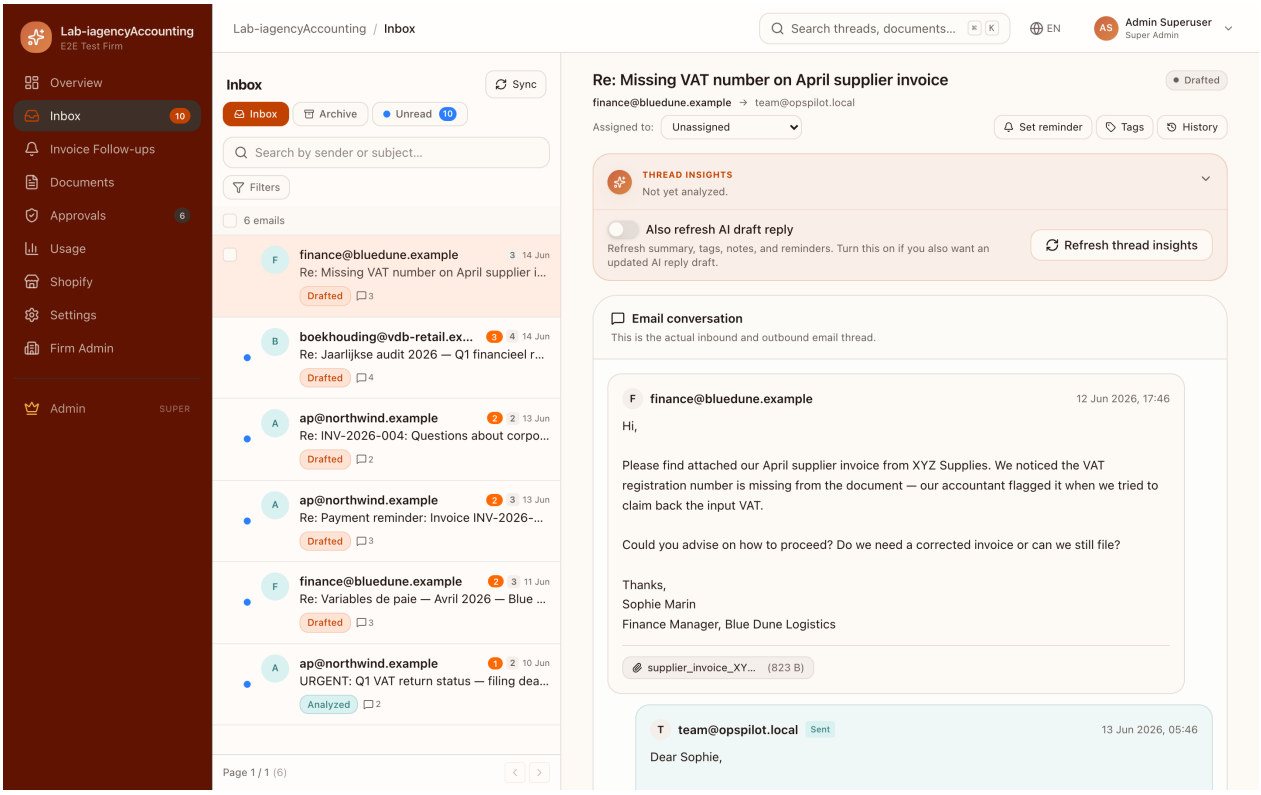

Figure 4 Thread view: AI-generated thread insights summarise the client's request, detected workflow tags, and the next action. The right panel hosts the AI draft, a TipTap-based rich editor, and one-click approval controls.

Thread grouping is done in a single GROUP-BY query (`EmailListView`) ordered before aggregation — skipping `.order_by()` caused Django to include `received_at` in the GROUP BY and break deduplication, a subtle bug that cost two days to track down and is now load-bearing.

4.4 Approvals

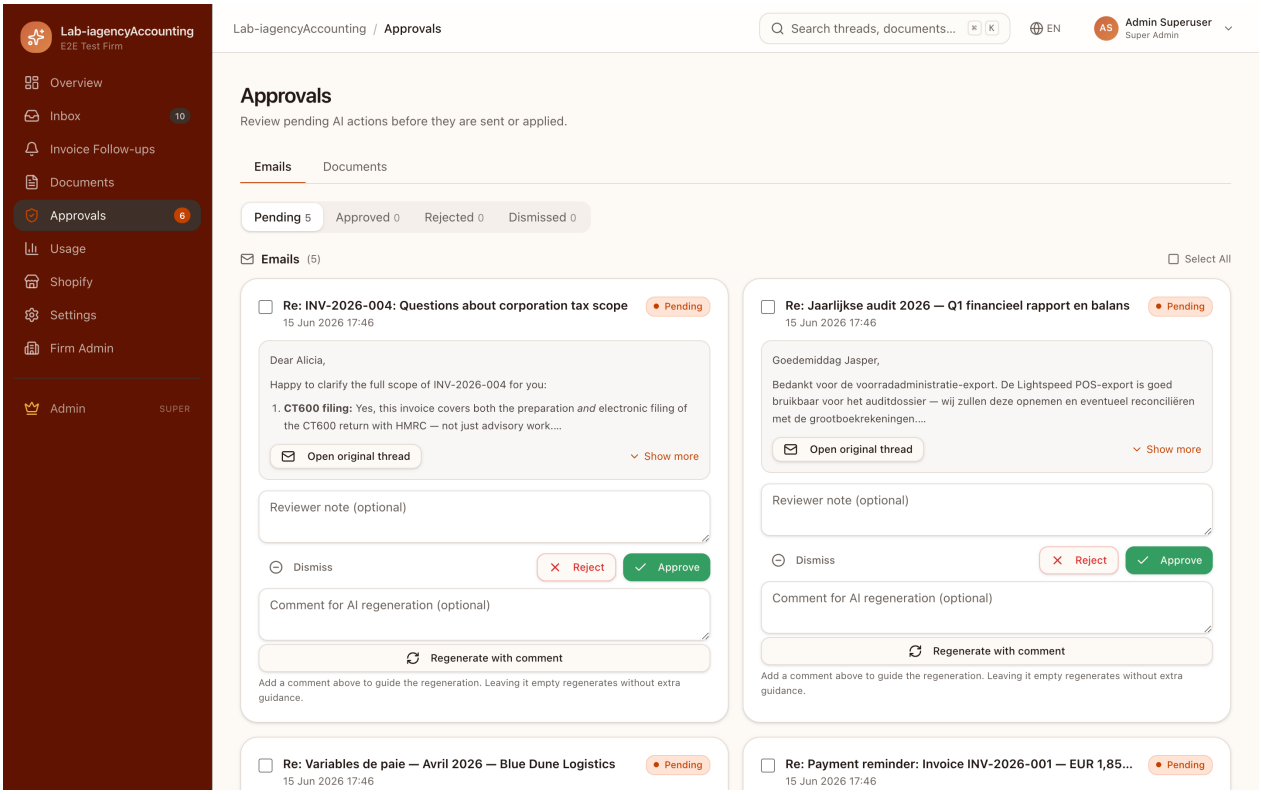

Figure 5 Approval queue: each card holds the AI draft, a reviewer note, a regeneration hint and the approve / reject / dismiss controls. Pending, approved, rejected and dismissed are separate tabs for audit purposes.

The approval model is generic: `ApprovalRequest` stores a target object reference, the proposed payload (HTML body, recipients, tags), and a UUID that is the URL slug. The same model also gates document classifications and outbound reminders — one workflow, three product surfaces.

4.5 Documents

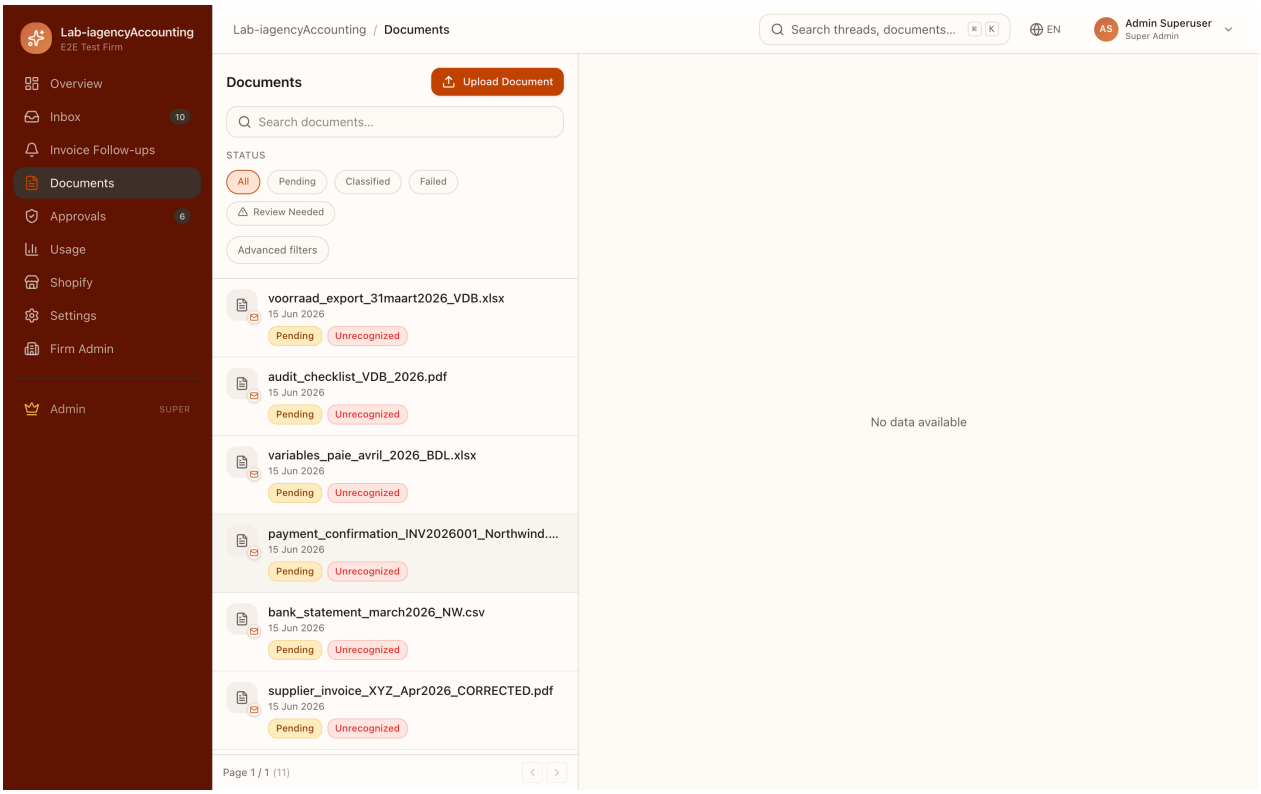

Figure 6 Document classification list. Each upload is scored, tagged (*Supplier Invoice*, *Unrecognized*, *Failed*, etc.) and routed for human review when confidence falls below threshold.

4.6 Reminders

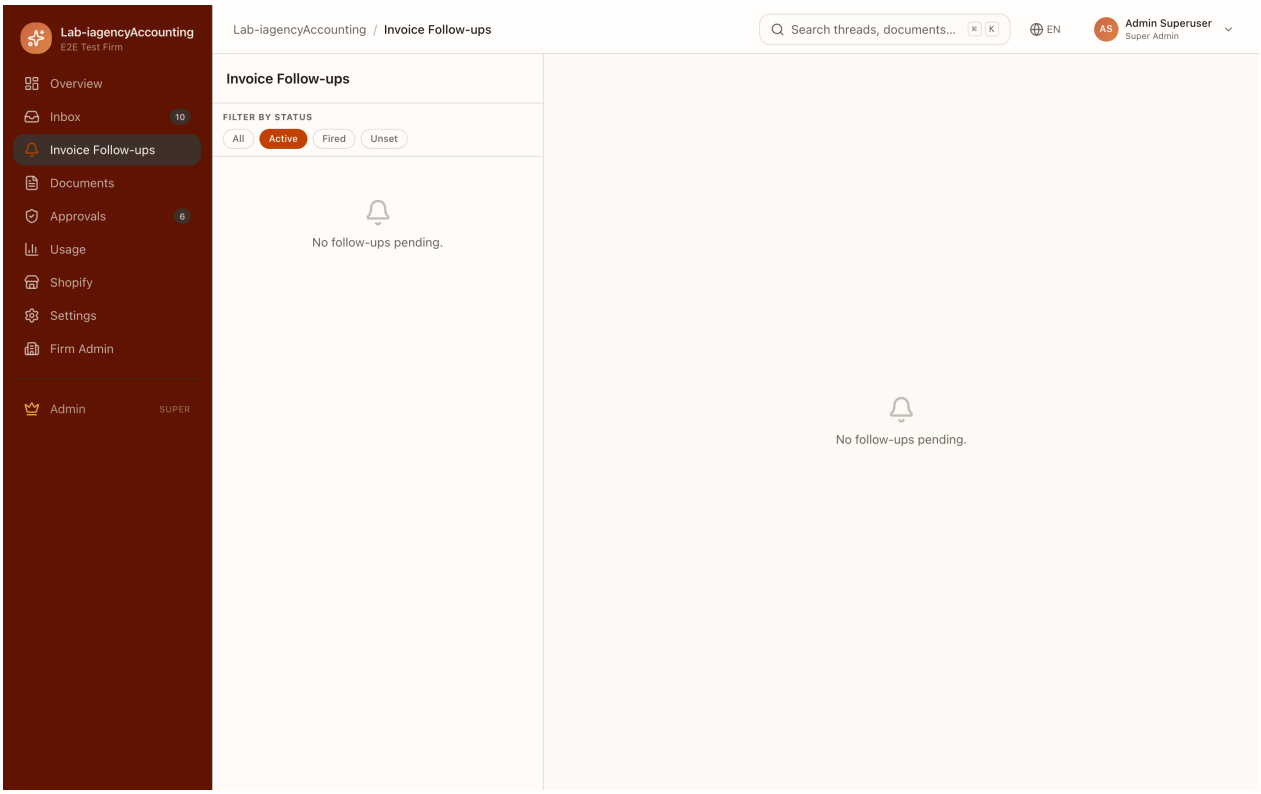

Figure 7 Scheduled follow-ups. Celery beat fires the reminder pipeline; reminders that need a generated message also pass through the approval queue.

4.7 Usage & cost tracking

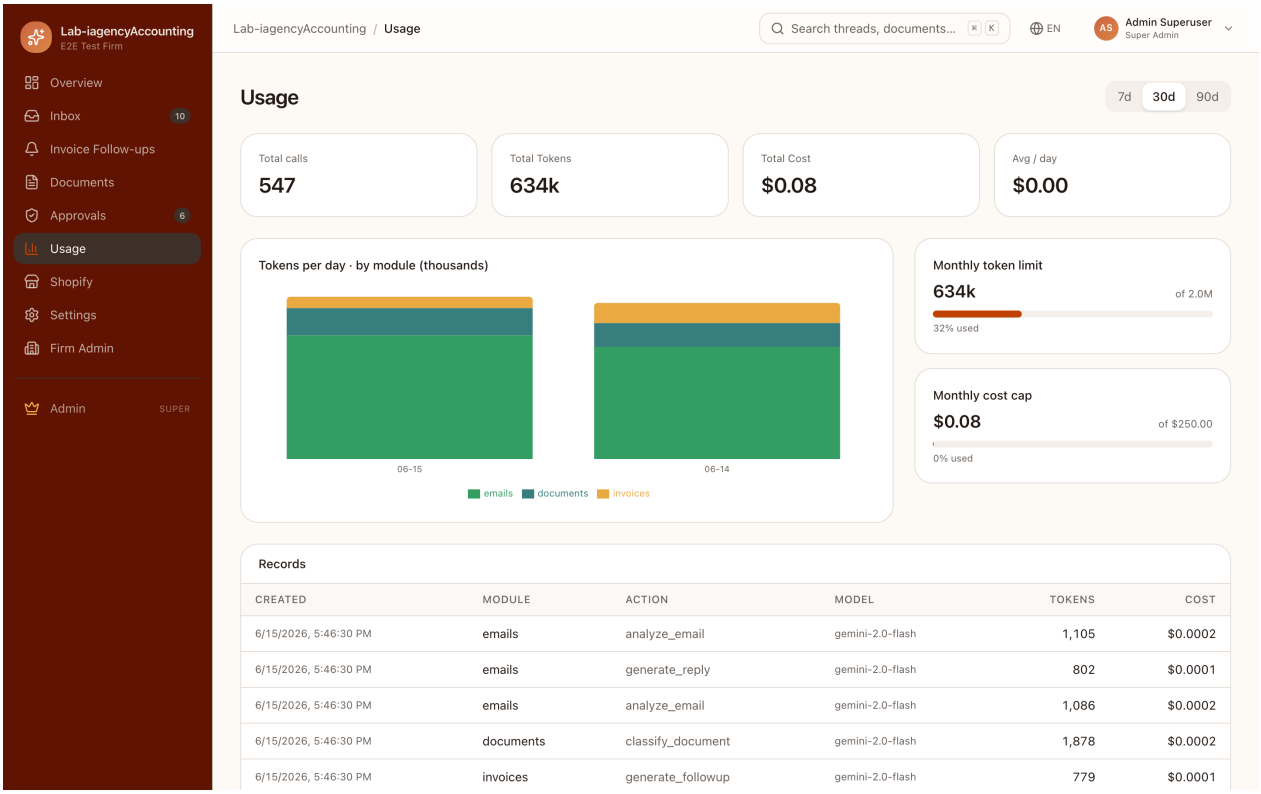

Figure 8 Per-firm AI usage: total calls, tokens, cost, and a daily bar chart by module. Monthly token and cost caps protect against runaway spend; the record log lists every call with model, action and exact cost.

Every Gemini call writes a row in `usage_tracking` with token counts and the firm's negotiated unit rate. The view is also the audit trail that auditors look at when answering "what did the AI do to that thread on May 18."

4.8 Settings

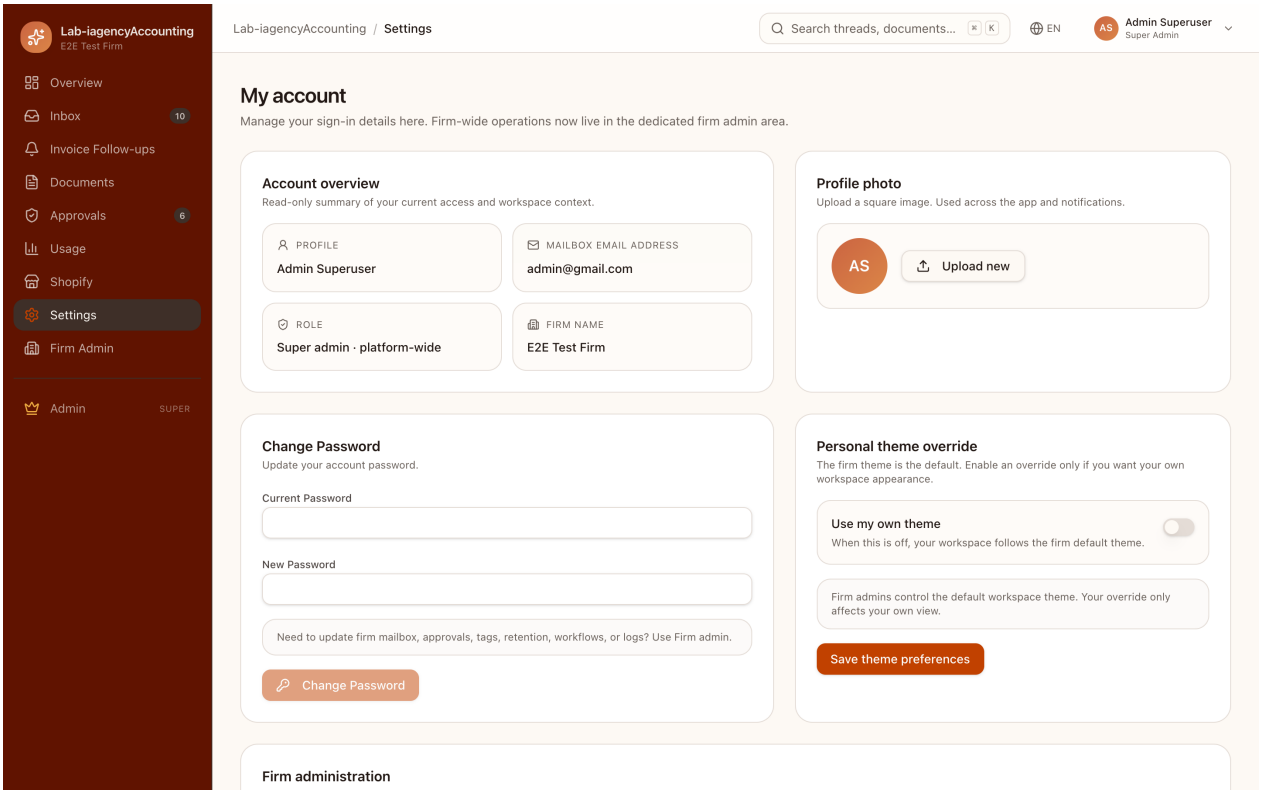

Figure 9 User-level settings: profile, language, notifications. The active firm and role are surfaced top-right so reviewers always know which tenant they are acting against.

4.9 Firm administration

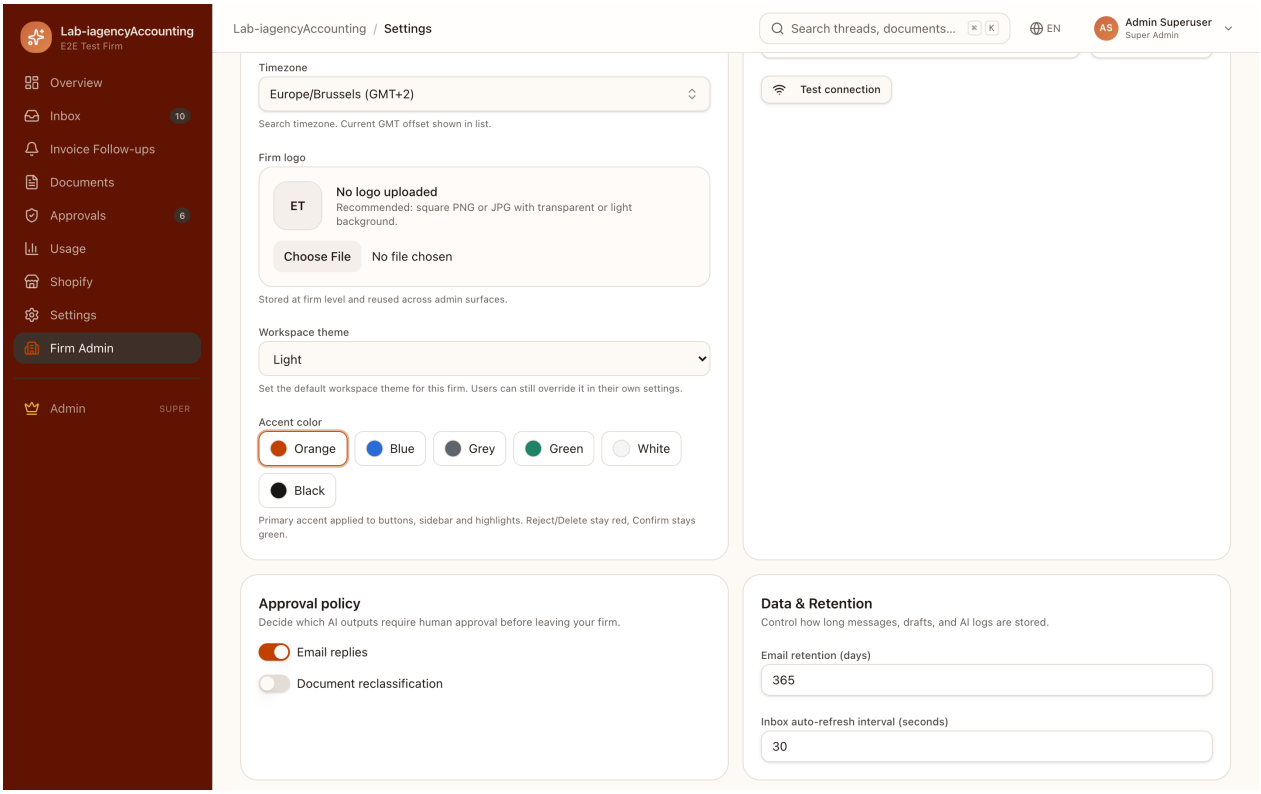

Figure 10 Firm-level admin: mailbox credentials (IMAP / SMTP, encrypted at rest), preferred language, email tone, accounting-software integration, plus tabs for tags, workflows, workflow rules and an audit log.

5 Tech Stack

5.1 Backend

Layer	Choice and rationale
Framework	Django 5.2 — mature ORM, admin, migrations, multi-tenant patterns
API	Django REST Framework + cookie JWT (SimpleJWT)
Async	Celery 5 with 4 named queues, gevent pool, 100 concurrency
Scheduler	Celery beat for IMAP polling and reminder dispatch
Database	Postgres 16
Broker / cache	Redis 7
AI	Google Gemini 2.5 Flash via `google-generativeai` SDK
Storage	Local filesystem in dev; S3-compatible in prod (`USE_S3_STORAGE`)
Auth	JWT cookies (access 1d, refresh 180d) + Google OAuth

5.2 Frontend

Layer	Choice and rationale
Framework	React 19 + TypeScript
Bundler	Vite 8 with HMR
UI primitives	MUI 6 + Radix UI + Tailwind 4 (utility layer on top of MUI tokens)
Data layer	TanStack Query 5 (no Redux / Zustand)
Forms	React Hook Form + Zod validation
Rich text	TipTap 3 (StarterKit, Underline, Placeholder) for HTML drafts
Charts	Recharts
i18n	react-i18next, EN / FR / NL locales
Routing	React Router 7 with `RequireAuth` / `GuestOnly` wrappers
Testing	Vitest, Testing Library, Playwright (e2e), MSW (network mocks)

5.3 Infrastructure

Component	Notes
Docker Compose	8 services, named network, persistent volumes for db / redis / media
Nginx	TLS termination, SPA hosting, reverse proxy to `aa-backend`
TLS	locally trusted certs via mkcert in dev; ACME in prod
Observability	Flower for Celery; structured JSON logs on the backend

5.4 Key engineering decisions

- **HTML-first drafts.** The AI is explicitly instructed to emit HTML (no markdown bold). `DraftBox` detects HTML vs plain-text drafts via a regex and either renders it directly or upgrades plain text into TipTap on edit. Legacy drafts written before this change still render correctly.
- **Approval as a first-class model.** Reusing one `ApprovalRequest` table across email replies, document classifications and reminders removed three near-identical pieces of approval code.
- **Thread grouping ordering.** `EmailListView` explicitly orders *before* the GROUP BY to keep Postgres from including `received_at` in the GROUP BY clause — a Django footgun documented in the codebase comments.
- **Container-scoped tooling.** Frontend installs require `--legacy-peer-deps` because Radix and MUI temporarily disagree on React 19 peers; this is captured in the project's onboarding docs so new contributors don't waste an afternoon on `npm install` failures.

6 Outcome

The platform replaces the most painful slice of the firm's day: the morning email triage. A licensed accountant who used to spend the first 90 minutes reading and replying now spends 15 minutes reviewing AI drafts, with a structured audit trail for every outbound message. Three languages, multi-tenant isolation, and complete cost visibility ship in the same release.

> Same headcount. Three times the client capacity. Nothing risky leaves the firm without a human approval.

7 Contact

Author	Serhat Keskin
Role	Full-stack engineer (Django / React / Celery / Docker)
Location	Available for remote engagements (CET / EU timezones)