MASTER PROMPT — WeaveSmart Chat (Chatwoot Fork)

> **READ FIRST — MANDATORY**
> You are an engineering assistant (Cursor / Trae / ProTry). **Follow this prompt strictly.** Do not add
> 1) Update or create a **README.md** (UK English).
> 2) **Commit** with a conventional message.
> 3) **Push** and (where applicable) open a PR.
> All code, UI copy, comments, docs, commits, and issues **must be in UK English**. Use **DD/MM/Y

---

0) Context & Goals
- Product: **WeaveSmart Chat (WSC)** — a **SaaS, multi■tenant, omnichannel** chat solution for U
- We will **leverage Chatwoot core** (conversations, inboxes, channels, messages) and add a **clear
- Two portals: **Tenant Dashboard** (client companies) and **Master Admin** (WeaveCode) — the la
- Channels: Web chat widget; **WhatsApp official (BSP)**; **WhatsApp third■party** (allowed but with
- Business model: Plans **Basic / Pro / Premium / App / Custom**, **no free plan**; **Stripe + PayPal
- Compliance: **UK GDPR**; strict **data isolation** by `tenant_id`; consent banner; retention policies
- Operations: Railway + Docker, **staging & production**, GitHub Actions, health checks, daily backup

Golden Rules
1) **Do not edit Chatwoot core** if a clean extension/concern/engine can achieve the same.
2) All **new APIs** live under our namespace and are documented with **OpenAPI**; generate a **Ty
3) All **new features** are protected by **feature flags** (per tenant and per plan).
4) Keep migrations **backwards■compatible** (expand/contract).
5) Meet **performance budgets** (see below).
6) Keep **security** strict: 2FA (owner/admin), CSP + SRI, rate limits, audit logs, encryption in transit,

---

1) Stack & Standards
- **Backend**: Ruby on Rails (Chatwoot base) + our engine `Weave::Core`.
- **Frontend**: Vue 3 + **Vuetify** (UI) + **Anime.js** (animations), i18n default **English (UK)**.
- **Database**: PostgreSQL; **Redis** for cache, queues, WebSocket.
- **Contracts**: OpenAPI for our endpoints; TS SDK auto■generated and imported by the web app.
- **Infra**: Docker **multi■stage**; Railway services (staging, prod).
- **CI/CD**: GitHub Actions; `develop` → staging; `main` → prod; release tags `vX.Y.Z`; CHANGELO
- **Git**: branches `main`, `develop`, `feature/*`, `fix/*`, `hotfix/*`; conventional commits.
- **Security & Ops**: 2FA (owner/admin), CSP + SRI, **rate■limiting per tenant/channel/module**, str
- **Design System**: Colours — primary `#8127E8`, accent `#FF6600`; fonts per WeaveCode brand;
- **Performance Budgets**:
  - **Web widget bundle ≤ 100KB gz**.
  - **Dashboard route bundle ≤ 200KB gz**.

- API **p95 ≤ 300ms** (read) / **≤ 600ms** (write) on staging data set.

---

2) Environments & Policies
- **Staging** and **Production** on Railway. Staging must be clearly marked in UI.
- Staging data **resettable on command**; seed scripts for demo tenants.
- Environment variables managed via Railway secrets; no secrets in code.
- **UK English** everywhere (UI, logs, docs).
- Date: **DD/MM/YYYY**; Time: 24h; Currency: **GBP (£)**.
- Third■party **WhatsApp unofficial** connections show a **non■dismissable banner** with a link to W

---

3) Definition of Ready / Done
- **Ready**: block scope is clear; acceptance criteria testable; feature flags identified; env vars define
- **Done**: code + tests (where applicable) green; budgets respected; README updated; conventiona

---

4) Tiny Work Blocks (each ends with README + commit + push)
> **Block template (apply to every block):**
> **Objective** — **Tasks (micro■steps)** — **Acceptance Criteria** — **README (UK English)** –

BLOCK 01 — Fork Preparation & Upstream Sync
Objective: Prepare fork, branches, basic CI lint.
Tasks: Fork Chatwoot; set `main` & `develop`; add `upstream`; add `.editorconfig`, `CODEOWNER
Acceptance: CI green; upstream fetch/merge documented.
README: branching, remotes, sync guide.
Versioning: `feature/bootstrap-fork` → `chore: bootstrap fork and CI lint`.

BLOCK 02 — Docker Multi■Stage & Railway Staging
Objective: Single container (Nginx + Rails + Node) and staging deploy.
Tasks: Author `Dockerfile` multi■stage; `docker-compose.yml` for local (Postgres/Redis); Railway
Acceptance: staging reachable; `/health` = 200.
README: local run + Railway deploy.
Versioning: `feature/docker-railway-staging` → `chore: add multi-stage Docker and staging deploy`

BLOCK 03 — Engine `Weave::Core` Namespace
Objective: Isolate our custom work from Chatwoot core.
Tasks: Create Rails engine; mount at `/api/weave/v1`; base controller with Chatwoot auth.
Acceptance: `GET /api/weave/v1/ping` returns `{ ok: true }`.
README: engine purpose and usage.

Versioning: `feature/weave-core-engine` → `feat: add Weave::Core engine and base route`.

BLOCK 04 — OpenAPI + Generated TS SDK
Objective: API■first contract for our endpoints.
Tasks: Add rswag; serve `/docs/openapi.json`; pipeline to generate TS SDK into web app.
Acceptance: web consumes generated SDK successfully.
README: how to update spec and regenerate SDK.
Versioning: `feature/openapi-sdk` → `feat: OpenAPI spec and generated TypeScript SDK`.

BLOCK 05 — Multi■Tenant by `tenant_id`
Objective: Enforce tenant isolation on new tables/services.
Tasks: Add `tenant_id` to our tables; composite indexes (`tenant_id, created_at`); service layer en
Acceptance: tests prove no cross■tenant leaks.
README: multi■tenant policy.
Versioning: `feature/tenant-scope` → `feat: tenant scoping and composite indexes`.

BLOCK 06 — Feature Flags (Tenant/Plan)
Objective: Toggle features without redeploy.
Tasks: Add Flipper/Unleash; model flags by tenant and by plan; helpers in code and guards in UI.
Acceptance: flags visible and effective; master admin can see toggles.
README: naming convention and usage.
Versioning: `feature/feature-flags` → `feat: tenant/plan feature flags`.

BLOCK 07 — Plans & Billing (Stripe/PayPal) + Trial
Objective: Basic/Pro/Premium/App/Custom, 7■day trial, auto■activation.
Tasks: Tables `plans`, `subscriptions`; Stripe & PayPal webhooks; entitlements via flags; “My Plan
Acceptance: upgrade/downgrade/cancel flow; trial expiry suspends access.
README: plan matrix and entitlements.
Versioning: `feature/billing` → `feat: plans, subscriptions and webhooks`.

BLOCK 08 — Master Admin (WeaveCode)
Objective: Manage tenants without reading their private data.
Tasks: List tenants (status, plan, trial, channels); actions: suspend, benefit, force upgrade; alerts fo
Acceptance: master cannot read tenant conversations.
README: admin scope and safeguards.
Versioning: `feature/master-admin` → `feat: master admin console`.

BLOCK 09 — RBAC per Tenant (Roles & Policies)
Objective: Owner/Admin, Agents, Finance, Support; customisable permissions.
Tasks: Define roles; policy checks backend; UI menu/action guards.
Acceptance: access matrix enforced by tests.
README: permissions table and examples.
Versioning: `feature/rbac` → `feat: tenant-scoped RBAC`.

BLOCK 10 — Permanent Risk Banner (WhatsApp Unofficial)
Objective: Non■dismissable banner + FAQ link + e■mail on switch to official.
Tasks: Detect `integration_type=unofficial`; render sticky banner; send confirmation e■mail when s
Acceptance: cannot hide banner until official API is used.
README: legal/disclaimer copy.
Versioning: `feature/wa-unofficial-warning` → `feat: persistent risk banner for unofficial WhatsApp`

BLOCK 11 — Rate Limiting by Tenant/Channel/Module
Objective: Control costs and stability.
Tasks: Implement per■plan rate caps; return 429 with friendly UI messages; admin config page.
Acceptance: burst tests and cooldown verified.
README: plan policies and overrides.
Versioning: `feature/rate-limits` → `feat: plan-based rate limiting`.

BLOCK 12 — SLA, Departments & Queues
Objective: Service targets and routing.
Tasks: Models `departments`, `queues`, SLA targets; alerts on breach; reporting hooks.
Acceptance: SLA metrics visible in reports.
README: how to configure SLA per queue.
Versioning: `feature/sla-queues` → `feat: departments, queues and SLA targets`.

BLOCK 13 — Advanced Search
Objective: Powerful filters with fast indexes.
Tasks: Filters by channel/agent/tags/dates/text; proper indexes; search UI.
Acceptance: p95 ≤ 300ms on staging data.
README: indexed fields and usage examples.
Versioning: `feature/advanced-search` → `feat: advanced search and indexing`.

BLOCK 14 — Append■Only Event Log + Batch Aggregations
Objective: Analytics without slowing hot path.
Tasks: Table `weave_events` (append■only); hourly jobs to materialised report tables.
Acceptance: dashboards read from aggregates.
README: event schema and jobs.
Versioning: `feature/event-log` → `feat: append-only event log and batch aggregations`.

BLOCK 15 — Tenant Reports + CSV/PDF Exports
Objective: Visual insights with period filters.
Tasks: Cards & charts: volume, sources, FRT/ART, closed/abandoned; exports CSV/PDF.
Acceptance: filters and downloads work.
README: KPI definitions.
Versioning: `feature/reports-tenant` → `feat: tenant reports and exports`.

BLOCK 16 — Master Reports (Platform Governance)
Objective: Fleet■level usage and alerts.
Tasks: Aggregates by tenant/plan; channel status; incident list.
Acceptance: no sensitive data loaded.
README: platform metrics explained.
Versioning: `feature/reports-master` → `feat: master-level reports`.

BLOCK 17 — Consent, CSAT & NPS
Objective: GDPR consent + post■chat satisfaction.
Tasks: Widget consent banner; **CSAT (1–5)**; optional **NPS** (per plan); reporting.
Acceptance: results appear in reports.
README: triggers and data fields.
Versioning: `feature/consent-csat-nps` → `feat: consent banner, CSAT and optional NPS`.

BLOCK 18 — Data Retention & Purge per Tenant
Objective: Configurable policies with audit trail.
Tasks: Retention settings per tenant (e.g., 12/24/36 months); scheduled purge job; audit logs.
Acceptance: old records removed as configured.
README: configuring retention.
Versioning: `feature/retention-purge` → `feat: tenant data retention and purge jobs`.

BLOCK 19 — Two■Factor Authentication (Owner/Admin)
Objective: Strengthen privileged access.
Tasks: TOTP 2FA; recovery flows; enforcement for owner/admin.
Acceptance: 2FA tests pass; UX validated.
README: policy and support instructions.
Versioning: `feature/2fa` → `feat: two-factor authentication for privileged roles`.

BLOCK 20 — CSP, SRI & Bundle Budgets
Objective: Frontend security and performance.
Tasks: Strict CSP headers; SRI for bundles; ensure widget ≤ 100KB gz and each dashboard route
Acceptance: Lighthouse and manual checks pass.
README: policies and measuring steps.
Versioning: `feature/csp-sri-budgets` → `feat: CSP, SRI and bundle budgets`.

BLOCK 21 — Observability (Sentry + Prometheus)
Objective: Production telemetry.
Tasks: Sentry (front/back); Prometheus exporters for Rails/Sidekiq; `/metrics`, `/health`, `/ready`.
Acceptance: dashboards show metrics and errors.
README: running and reading metrics.
Versioning: `feature/observability` → `feat: Sentry and Prometheus exporters`.

BLOCK 22 — Backups (Daily) + Weekly Restore Test + (Optional) PITR

Objective: Reliable disaster recovery.
Tasks: Automate daily DB/files backups; weekly restore test to temp DB; document PITR option wh
Acceptance: restore test succeeds with logs.
README: disaster recovery playbook.
Versioning: `feature/backups-restore-test` → `feat: daily backups and weekly restore test`.

BLOCK 23 — Omnichannel Connectors (Skeleton + Diagnostics)
Objective: WhatsApp/Telegram/Facebook/Instagram/E■mail integrations.
Tasks: Connection screens; credentials storage (masked); **Send test** and **Health** diagnostic
Acceptance: test sends work for each channel.
README: how to connect; limits and costs.
Versioning: `feature/channel-connectors` → `feat: omnichannel connectors with diagnostics`.

BLOCK 24 — Basic Automation (OOH, Welcome, Quick Replies)
Objective: Non■AI rules.
Tasks: Out■of■hours replies; welcome messages; canned responses; conversation logs.
Acceptance: can enable/disable via flags.
README: rule catalogue.
Versioning: `feature/automation-base` → `feat: basic automation rules`.

BLOCK 25 — AI Core (Intent, Sentiment, Summary, Reply Suggest)
Objective: Flag■controlled AI for higher plans.
Tasks: Pluggable providers (OpenAI, etc.); endpoints + UI switches; PII sanitisation in logs.
Acceptance: visible results; per■plan limits enforced.
README: keys, quotas, privacy notes.
Versioning: `feature/ai-core` → `feat: AI intent/sentiment/summary/reply suggestions`.

BLOCK 26 — Attachments & S3■Compatible Storage
Objective: Signed URLs and tenant■scoped prefixes.
Tasks: Local provider for dev; S3/R2 for prod; table `attachments` with checksums.
Acceptance: upload/download with expiry.
README: env vars and rotation.
Versioning: `feature/storage-adapter` → `feat: S3-compatible storage adapter`.

BLOCK 27 — Scheduled Messages
Objective: Future delivery with retry/backoff.
Tasks: Model scheduling by conversation/channel; worker with retry; cancellation flow.
Acceptance: reliable execution; audit logs.
README: limits per plan.
Versioning: `feature/scheduled-messages` → `feat: scheduled messages`.

BLOCK 28 — Internal Notes, @Mentions & Reactions
Objective: Agent collaboration.

Tasks: Private notes; mentions with notifications; simple reactions; permissions.
Acceptance: role checks enforced.
README: events and UI behaviours.
Versioning: `feature/collab-notes` → `feat: internal notes, mentions and reactions`.

BLOCK 29 — Lightweight Web Widget
Objective: Custom theme, consent, one■line embed.
Tasks: Build UMD/ESM; theme tokens (WeaveCode colours); consent banner; i18n default UK.
Acceptance: ≤ 100KB gz; basic E2E works.
README: embed snippet and options.
Versioning: `feature/web-widget` → `feat: lightweight embeddable web widget`.

BLOCK 30 — CI/CD (Staging→Prod) + Health Gates & Rollback
Objective: Trustworthy releases.
Tasks: Actions: PR build/test; `develop`→staging; `main`→prod; post■deploy health; automatic ro
Acceptance: reproducible deployments.
README: release & rollback flows.
Versioning: `feature/cicd-release` → `chore: full CI/CD with health gates and rollback`.

BLOCK 31 — Apple Messages for Business (Placeholder)
Objective: Prepare structure and flags; no hard dependency.
Tasks: Feature flag; contract stub; UI placeholder; docs of requirements.
Acceptance: safely off by default.
README: enablement steps.
Versioning: `feature/apple-business-messages` → `chore: placeholder for Apple Messages for Bus

BLOCK 32 — Google Business Messages (Placeholder)
Objective: Same pattern as Block 31.
Tasks: Flag + stub + docs.
Acceptance: off by default.
README: prerequisites.
Versioning: `feature/google-business-messages` → `chore: placeholder for Google Business Mess

BLOCK 33 — Knowledge Base / Help Centre
Objective: Multi■language help with semantic search.
Tasks: KB module; categories, articles; search; permissions; widget integration.
Acceptance: renders in widget and admin.
README: authoring and search notes.
Versioning: `feature/help-centre` → `feat: knowledge base and help centre`.

BLOCK 34 — Proactive In■App Tours & Nudges
Objective: Journeys, hints, checklists.
Tasks: In■app prompts; targeting rules; scheduling.

Acceptance: per■tenant targeting works.
README: targeting rules.
Versioning: `feature/proactive-tours` → `feat: proactive in-app tours and nudges`.

BLOCK 35 — WhatsApp Commerce (Interactive Lists/Payments)
Objective: Catalogue flows via WhatsApp templates.
Tasks: Support interactive messages; payments flow; compliance notes.
Acceptance: end■to■end demo.
README: limitations per BSP.
Versioning: `feature/wa-commerce` → `feat: WhatsApp commerce interactive flows`.

BLOCK 36 — Omnichannel Journey/Campaign Builder
Objective: Event■driven journeys across channels.
Tasks: Visual builder; triggers; actions (email/in■app/WhatsApp); throttling.
Acceptance: sample journey runs.
README: blocks catalogue.
Versioning: `feature/journey-builder` → `feat: omnichannel journey builder`.

BLOCK 37 — Marketplace & One■Click Connectors
Objective: App store for connectors (Shopify, Woo, HubSpot, Zoho, etc.).
Tasks: Install/uninstall; OAuth where needed; billing hooks.
Acceptance: connector lifecycle works.
README: connector template.
Versioning: `feature/marketplace` → `feat: marketplace and one-click connectors`.

BLOCK 38 — Mobile SDK (Embed in Client Apps)
Objective: Lightweight SDK to embed WSC in iOS/Android apps.
Tasks: Create mobile SDK repo; auth; widget bridge; docs.
Acceptance: sample app integrates.
README: integration steps.
Versioning: `feature/mobile-sdk` → `feat: mobile SDK for iOS/Android`.

BLOCK 39 — Voice/IVR & Voice AI (Placeholder → MVP)
Objective: Prepare voice channel and AI connector.
Tasks: Flags; call events; basic IVR tree; transcription provider stub.
Acceptance: off by default; demo call flow works when enabled.
README: providers & costs.
Versioning: `feature/voice-ivr` → `feat: voice/IVR channel (MVP scaffold)`.

BLOCK 40 — Co■Browsing / Video Support
Objective: Real■time assistance features.
Tasks: WebRTC scaffolding; permissions; session logs; bandwidth guards.
Acceptance: demo session works.

README: privacy & consent.
Versioning: `feature/cobrowse-video` → `feat: co-browsing and video support`.

BLOCK 41 — Usage■Based Billing (Optional)
Objective: Bill by usage metrics where appropriate.
Tasks: Metering per module; invoice items; caps and alerts.
Acceptance: invoices reflect usage.
README: metrics & caps.
Versioning: `feature/usage-billing` → `feat: optional usage-based billing`.

BLOCK 42 — Identity Resolution (Multi■Channel Profiles)
Objective: Merge contacts across channels.
Tasks: Deterministic + probabilistic rules; review queue; audit.
Acceptance: merges logged; manual override possible.
README: rules & risks.
Versioning: `feature/identity-resolution` → `feat: identity resolution across channels`.

BLOCK 43 — Agent Copilot (RAG)
Objective: Knowledge■assisted agent replies.
Tasks: Connectors (Drive/Confluence/Notion); embeddings; retrieval; UI pane.
Acceptance: suggested answers with sources.
README: data handling & privacy.
Versioning: `feature/agent-copilot` → `feat: agent copilot (RAG) with connectors`.

BLOCK 44 — OpenTelemetry Tracing
Objective: Distributed traces end■to■end.
Tasks: OTEL SDK; propagate `traceId`; export to chosen backend.
Acceptance: traces visible across services.
README: how to inspect.
Versioning: `feature/otel-tracing` → `chore: OpenTelemetry tracing`.

BLOCK 45 — Partitioning & UUIDv7 (If/When Needed)
Objective: Scale hot tables (messages/events).
Tasks: Use UUIDv7 for new IDs; monthly partitions for messages; migration notes.
Acceptance: partitions created; perf baseline improved.
README: when to partition and how.
Versioning: `feature/partitioning-uuidv7` → `chore: UUIDv7 and table partitioning`.

BLOCK 46 — Rate■Limit Dashboards & Admin Controls
Objective: Visualise and adjust caps safely.
Tasks: Admin UI; presets per plan; audit changes.
Acceptance: changes apply immediately.
README: governance notes.

Versioning: `feature/rate-limit-admin` → `feat: rate limit dashboards and controls`.

BLOCK 47 — GDPR DSR Toolkit (Export/Delete)
Objective: Fulfil data subject requests.
Tasks: Export packages; delete workflow with grace period; audit trail.
Acceptance: verified on staging.
README: operator guide.
Versioning: `feature/gdpr-dsr` → `feat: GDPR export and delete toolkit`.

BLOCK 48 — Operational Runbooks & Admin Scripts
Objective: Smooth operations for support engineers.
Tasks: Scripts for reset 2FA, block user, reseed staging, rotate keys; runbooks for incidents.
Acceptance: documented and tested.
README: runbook index.
Versioning: `feature/runbooks` → `docs: operational runbooks and scripts`.

BLOCK 49 — Staging Data Seed & Reset Command
Objective: Consistent demo/testing state.
Tasks: Seed tenants, agents, sample conversations; CLI reset; banner "Staging" in UI.
Acceptance: one■command reset works.
README: seeding policy.
Versioning: `feature/staging-seed` → `chore: staging seed and reset tooling`.

BLOCK 50 — Security Hardening & Go■Live Checklist
Objective: Final pass before GA.
Tasks: Pen■test fixes; headers review; rate■limits; secrets rotation; backups verified; restore test
Acceptance: checklist signed off.
README: go■live checklist stored in repo.
Versioning: `feature/go-live` → `chore: security hardening and go-live checklist`.

---

5) README Standard (apply to every block)
In **UK English**, each module README must include:
- **Overview** (what & why)
- **Setup** (env vars, flags)
- **How to test** (commands, expected outputs)
- **Limitations & Next steps**
- **Security & Privacy notes** (if applicable)

---

6) Commit / PR Standard

- **Branch**: `feature/<kebab-case>` or `fix/<kebab-case>`
- **Conventional commits**: `feat: …`, `fix: …`, `chore: …`, `docs: …`, `refactor: …`
- **PR Description**: what changed, why, how to test, risks, roll■back plan
- **Releases**: tag `vX.Y.Z`; update **CHANGELOG**.

---

7) Notes & Policies Recap
- **Language**: prompt in Portuguese earlier for planning; **this master prompt is in UK English** for e
- **Risky integrations** (WhatsApp third■party): permanent banner + FAQ link; cost borne by the clien
- **Privacy**: logs must avoid PII; where unavoidable, apply minimisation and masking.
- **Testing**: prefer small E2E where critical; unit/integration where practical; keep pipelines fast.
- **Extensibility**: new blocks can be added using the same template without breaking the MVP.

END OF MASTER PROMPT — Execute blocks sequentially. Every block **must** finish with **RE