Content Generation System

Assumptions

- We'll use the existing NestJS API (`apps/api/src`), Node worker (`apps/worker/worker.js`), Redis-backed queues (`queue.service.ts`), and Prisma DB (`packages/db/prisma/schema.prisma`).
- Perplexity API will be used for retrieval-augmented generation and fact-checking; OpenAI/Groq-like LLM can be swapped in via provider interface.
- SEO, quality, and platform-adaptation are enforced as services that plug into existing `content-adaptation.service.ts` and `platform-publish.service.ts`.
- spaCy will run in a separate Python microservice or via `spacy-js` compatible endpoints for performance and model availability.

Module Goals (Week 4–5)

- Content Generation System: Prompt templates, workflows, SEO optimization, variation engine, quality checks.
- NLP & Engagement Module: Intent detection, rules engine, lead extraction, escalation triggers, sentiment.

Architecture Overview

- API: Orchestrates jobs and exposes endpoints to create/generate/validate/publish content.
- Worker: Executes long-running content-gen workflows, NLP processing, validation, and publishing.
- Services:
  - `PerplexityService` for RAG/fact-check.
  - `ContentGenerationService` for templating + LLM calls.
  - `SeoOptimizationService` for keywords, structure, links, readability.
  - `VariationService` for channel-specific adaptation.

    - `QualityValidationService` for safety, plagiarism, factuality, PII.
    - `NlpService` (spaCy gateway) for intents, NER, sentiment.
    - `EngagementRulesService` for rule-based responses and escalation.
    - `LeadExtractionService` for contact/lead capture.
- Data: Prisma models for Content, Variations, SEO, QualityReport, EngagementEvent, Leads, and Audit.
- Queues: `content.generate`, `content.validate`, `content.adapt`, `content.publish`, `nlp.process`, `engagement.respond`, `lead.extract`.

Detailed Plan

1) Perplexity Integration with Prompt Templates

- Add `PerplexityService` wrapper:
    - Methods: `search(query, options)`, `answer(question, context, options)`, `factCheck(claims[], k)`.
    - Config via `packages/config/env.ts` (`PERPLEXITY_API_KEY`, base URL, model).
- Prompt Templates:
    - Store templates in DB (`PromptTemplate` table) with fields: `name`, `version`, `inputSchema`, `template`, `outputSchema`, `channel`.
    - Add helper: `renderTemplate(template, variables)` with validation against `inputSchema`.
- Retrieval Flow:
    - For blog/newsletter: 1) collect brief + angle; 2) `PerplexityService.search` for sources; 3) synthesize outline; 4) draft; 5) citations; 6) fact-check pass.
- Caching:
    - Cache Perplexity search results by `(queryHash, dateBucket)` in Redis for 24h to reduce cost.
- Observability:
    - Emit spans: `perplexity.search`, `perplexity.answer`, `perplexity.factcheck` with token counts and latency.

2) Content Generation Workflows (Blogs/Newsletters)

- Define workflow states in DB: `DRAFTING → SEO_ENHANCED → VALIDATED → ADAPTED → PUBLISHED`.
- Worker jobs:
  - `content.generate`: render prompt, call LLM, attach sources, save draft.
  - `content.seo`: enrich with keywords, H-tags, meta, links.
  - `content.validate`: quality checks (toxicity, plagiarism, fact-check score).
  - `content.adapt`: variations per platform and length.
  - `content.publish`: hand off to `platform-publish.service.ts`.
- API endpoints:
  - `POST /content/generate` body: topic, targetAudience, tone, channels, type: blog/newsletter.
  - `GET /content/:id` to view content with SEO and quality reports.
  - `POST /content/:id/publish` with target channels, schedule.
- Idempotency:
  - Use existing `idempotency.middleware.ts` for generation POSTs.

3) SEO Optimization Algorithms

- Keyword discovery:
  - Use Perplexity for SERP-like topical clusters; extract candidate keywords + volumes (if available).
  - Rank keywords by TF-IDF over draft + source coverage + competitor mention frequency.
- Structure:
  - Enforce H1 (single), H2/H3 hierarchy; ensure paragraph lengths, table-of-contents for blogs >1200 words.
- On-page meta:
  - Title length 50–60 chars, meta description 140–160 chars.

    - Slugify titles; internal link placeholders to existing content via ES index (`test/es_content_index.json` prototype).
- Readability:
    - Compute Flesch-Kincaid grade; enforce target by audience; suggest rephrases.
- Links:
    - Validate external links with 200 status in worker; anchor text variation.
- Output:
    - SEO report entity: `score`, `keywords[]`, `readability`, `structureWarnings[]`, `meta`, `links[]`.

4) Content Variation Engine (Multi-Platform)

- Channel adapters:
    - LinkedIn: 300–700 chars post + 1–3 hooks, 3–5 hashtags; CTA optional.
    - X/Twitter: 1–2 tweets per thread segment, 1–2 hashtags, mention style.
    - Instagram: 125–150 chars primary, line breaks, 10–15 hashtags, alt text.
    - Facebook: 1–2 short paragraphs, link preview optimization.
    - Newsletter (Email): subject lines (A/B), preview text, sections, buttons.
- Constraints:
    - Add per-platform max length, emoji usage, hashtag patterns, link policy.
- Media:
    - Use `services/media-processing.service.ts` to propose crop/alt text based on content summary.
- A/B Variations:
    - Generate 2–3 variants per channel with different hooks/CTAs; track selection and performance fields.

5) Content Quality Checks and Validation

- Safety/Toxicity:

    - Use an open model (e.g., Detoxify API) or LLM moderation endpoint; score threshold and rationale.
- Plagiarism:
    - Shingling + cosine similarity vs cached web snippets from Perplexity sources; highlight overlaps.
- Factuality:
    - Extract claims from draft; `PerplexityService.factCheck` with K evidence; compute support score.
- Style Guide:
    - Lint for passive voice, sentence length, jargon; brand voice toggles (tone, forbidden words).
- PII detection:
    - Regex + NER for emails, phone, addresses; mask or flag.
- Output:
    - `QualityReport` with sub-scores, failing items, auto-fix suggestions.

6) NLP & Engagement Module

6.1 spaCy-based Intent Detection

- Deploy `en_core_web_trf` or `en_core_web_md` depending on latency budget.
- Create `nlp-service` (Python) endpoint:
    - `POST /nlp/analyze` → intents (textcat), entities (NER), sentiment proxy, language.
- Training:
    - Seed intents: lead inquiry, pricing, support, booking, feedback, general chit-chat.
    - Store training data in DB for continuous learning; allow CSV import.
- API:
    - `NlpService` in API calls the Python service; add circuit-breaker and timeouts.

6.2 Rule-based Response Engine

- YAML-driven rules:
  - Conditions: intent, sentiment range, entities present, channel, customer tier.
  - Actions: send template response, ask clarification, escalate, capture lead, schedule follow-up.
- Templating with variables from NER; throttle and dedup rules with idempotency keys.
- Test harness: simulate inputs, assert chosen rule and response.

6.3 Lead Extraction Algorithms

- Combine spaCy NER (PERSON, ORG, EMAIL, PHONE) + regex validation + heuristics (signature blocks, URLs).
- Normalize using libphonenumber, email DNS MX check async.
- Store `Lead` with source channel, confidence, lastMessage, extractedAt.

6.4 Escalation Triggers

- Conditions:
  - Low confidence intent, negative sentiment below threshold, VIP account, repeated unresolved thread, legal/compliance keywords.
- Actions:
  - Create `EscalationTicket`, notify Slack/MS Teams via webhook, assign owner, SLA timer.

6.5 Sentiment Analysis

- Lightweight: VADER or `textblob` at edge; escalations rely on negative thresholds.
- Richer fallback: transformer sentiment API if message length > N or uncertainty high.
- Aggregate conversation sentiment trend per thread.

Data Model Additions (Prisma)

- `Content`: id, type, topic, audience, tone, draft, outline, citations[], state, createdBy, createdAt.
- `ContentSeo`: contentId, score, keywords[], readability, metaTitle, metaDescription, internalLinks[], externalLinks[].
- `ContentVariation`: contentId, channel, variantIndex, text, media[], constraints, selected, publishedAt, metrics?.
- `QualityReport`: contentId, safetyScore, plagiarismScore, factualityScore, styleScore, piiFindings[], issues[].
- `EngagementEvent`: id, channel, threadId, messageId, text, intent, sentiment, entities, handledBy, actionTaken.
- `Lead`: id, name?, email?, phone?, org?, sourceChannel, sourceMessageId, confidence, status.
- `EscalationTicket`: id, reason, severity, assignedTo?, createdAt, resolvedAt?, notes[].
- `PromptTemplate`: id, name, version, channel?, inputSchema, template, outputSchema, createdBy.

Queues and Workers

- Add queues: `content.generate`, `content.seo`, `content.validate`, `content.adapt`, `content.publish`, `nlp.process`, `engagement.respond`, `lead.extract`, `escalation.raise`.
- Update `apps/worker/worker.js`:
  - Register processors per queue with concurrency controls.
  - Use correlation IDs via `correlation.middleware.ts`.
  - Emit metrics for job latency, success/failure, retries.

Endpoints and Services (API)

- `POST /content/generate` → enqueue generate + return contentId.
- `GET /content/:id` → include SEO, quality, variations.

- `POST /content/:id/validate` → re-run validation.
- `POST /content/:id/adapt` → regenerate selected channels.
- `POST /content/:id/publish` → reuse `platform-publish.service.ts`.
- `POST /nlp/analyze` (proxy to Python nlp-service) for backoffice tools.
- `POST /engagement/ingest` → receive social messages/webhooks; enqueue `nlp.process`.

Observability and Governance

- Tracing spans per workflow stage; dashboards for throughput, avg gen time, fail rate, token cost.
- Audit logs for generated content changes and rule decisions.
- Feature flags for enabling channels, SEO strictness, quality gates.
- Rate limits on generation endpoints; idempotent keys to avoid duplicate drafts.

Testing and Acceptance

- Unit tests for templating, SEO, variation rules, rule-engine evaluation.
- Integration tests: end-to-end blog generation to publish (mock platform clients).
- NLP eval set with labeled intents; measure precision/recall > 0.85.
- Quality gate thresholds (configurable):
    - Safety ≥ 0.9, Plagiarism ≥ 0.85 (lower is worse), Factuality ≥ 0.8, Readability within target grade ±1.

Security and Compliance

- Secrets via `packages/config/env.ts` and deployment secrets manager.
- PII handling with masking in logs; opt-in storage for leads; data retention policy.
- Signed webhooks; platform tokens stored via existing `token.service.ts` with rotation.

Week-by-Week Breakdown

Week 4

- Perplexity integration and templates
- Blog/newsletter generation workflow and DB models
- SEO optimization service + report
- Quality validation service (toxicity, plagiarism baseline, fact-check V1)
- APIs for generate/fetch; worker queues wired

Week 5

- Variation engine for LinkedIn, X, Instagram, Facebook, Newsletter
- spaCy NLP service (deploy, wire intents/NER/sentiment)
- Rule-based response engine + escalation triggers
- Lead extraction pipeline and storage
- E2E tests, dashboards, and rollout flags

Risks and Mitigations

- External API limits/costs: add caching and backoff; budget alerts.
- spaCy latency: choose `md` model by default; batch process; cache NER for repeated texts.
- Plagiarism false positives: tune shingle size; whitelist citations.
- Factuality: clarify that score is advisory; require human review for low scores.
- Platform policy changes: isolate channel rules in config with versioning.

Deliverables

- Services implemented and wired to worker.
- New DB migrations for models above.
- API endpoints documented in `doc/postman-collection.json`.

- Dashboards/alerts per `doc/observability.md` patterns.
- Runbook updates in `doc/runbook.md` for failures and manual overrides.
- I can start implementing by scaffolding the services and DB models next.

Roadmap to integrate the Content Engine

Scope

End-to-end roadmap to integrate the Content Engine with social platforms and auto-post across LinkedIn, Facebook/Instagram (Meta), and X/Twitter, using your existing `platform-publish.service.ts`, `platforms/*-client.service.ts`, worker, queues, and scheduler.

Phase 0 — Foundations (Access, Models, Queues)

- **App registrations (scopes)**:
  - **LinkedIn**: w_member_social, r_organization_social, r_organization_admin, w_organization_social.
  - **Meta** (FB/IG): pages_manage_posts, pages_read_engagement, pages_show_list, instagram_basic, instagram_content_publish.
  - **X/Twitter**: tweet.read, tweet.write, users.read, offline.access.
- **OAuth + tokens**:
  - Use `oauth.controller.ts` + `token.service.ts` to obtain and store long-lived tokens and refresh flows per platform; capture accounts/pages to a `ConnectedChannel` table.
- **Data model**:
  - `Content`, `ContentVariation` (per channel), `ScheduleMetadata` (time zone, scheduleAt, recurrence), `PublishResult` (platformPostId, status), `ConnectedChannel`.
- **Queues**:

    - `content.adapt`, `content.publish`, `content.publish.retry`, `metrics.fetch`.
- **Rate limit/backoff**:
    - Centralized limiter in `platform-publish.service.ts` (per-platform quotas, exponential backoff, jitter).

Phase 1 — Channel Adapters and Validation

- **Channel adapters** (already scaffolded):
    - Extend `platforms/linkedin/linkedin-client.service.ts`, `platforms/meta/meta-client.service.ts`, and add `platforms/x/x-client.service.ts`.
    - Normalize API: `publishText`, `publishMedia`, `deletePost`, `lookupPost`, `scheduleIfSupported`.
- **Content validators**:
    - Per-channel checks: max length, hashtags, link policy, media aspect/size, alt text presence, mentions format.
- **Media pipeline**:
    - Reuse `services/media-processing.service.ts` to transcode/crop/compress; upload via channel pre-upload endpoints where required (e.g., IG container, X media upload).

Phase 2 — Orchestration and Scheduling

- **API**:
    - `POST /content/:id/publish` accepts `channels[]`, `scheduleAt`, `accountIds[]`, `dryRun`.
    - `GET /content/:id/publish-status` aggregates per-channel results.
- **Worker flow**:
    - `content.adapt`: create/update `ContentVariation` per selected channel.
    - `content.publish`:
        - Validate variation → prepare media → post via adapter → persist `PublishResult`.
    - `content.publish.retry`: handle transient errors (HTTP 429/5xx, network).

- **Scheduler**:
  - Use `apps/worker/cron.js` to scan future-dated `ScheduleMetadata` every minute; enqueue `content.publish`.
  - Time zone safe (store UTC, convert on input), idempotent keys to prevent double posts.

Phase 3 — Account Linking, Routing, and Approvals

- **Account routing**:
  - UI/API to list connected channels; map each `ContentVariation` to one or many `ConnectedChannel` entries.
- **Approval gate**:
  - State machine: `DRAFTING → SEO_ENHANCED → VALIDATED → ADAPTED → APPROVED → PUBLISHING → PUBLISHED/FAILED`.
  - Only `APPROVED` items can be scheduled/published (enforced in worker).
- **Compliance rules**:
  - Brand safelist/denylist terms, PII redaction for public channels, mention/handle validation.

Phase 4 — Posting Details Per Platform

- **LinkedIn**:
  - Choose entity (member vs organization). Create UGC Post with text + media, alt text; attach link if present; optional first-comment via second call.
- **Facebook**:
  - Page posts via Graph API; photo/video endpoints as needed; enable link preview; first-comment CTA optional.
- **Instagram**:
  - Use Container → Publish flow; handle reels vs feed; enforce caption limits, hashtag sets; alt text via `accessibility_caption`.

- **X/Twitter**:
    - OAuth 2.0; media upload chunks then `tweets` create; thread support by posting sequentially; hashtag limit/placement rules.

Phase 5 — Monitoring, Insights, and Webhooks

- **Status tracking**:
    - Store `platformPostId`, permalinks, publish timestamp, failure reason codes.
- **Webhooks/polling**:
    - Register platform webhooks where available (FB/IG, LinkedIn via notifications) or poll with backoff.
    - Ingest to `EngagementEvent`; update metrics snapshot fields on `PublishResult`.
- **Metrics**:
    - Impressions, likes, comments, shares, CTR where exposed; compute UTM-based site visits; nightly `metrics.fetch` jobs.

Phase 6 — Error Handling, Retries, and Rollbacks

- **Retry policy**:
    - 3 attempts with backoff for 429/5xx; no retry for 4xx validation errors (surface to user).
- **Partial failure**:
    - Post per-channel independently; persist granular statuses; allow manual re-publish per channel.
- **Rollback (delete)**:
    - Support `DELETE /publish/:id` to remove posts where API allows; log non-reversible cases.

Phase 7 — Security, Auditing, and Governance

- **Secrets**:
    - Keys in `packages/config/env.ts` + secret manager; never log tokens.

- **Auditing**:
  - Record who approved, when, and all payloads sent/received (with PII redacted).
- **RBAC**:
  - Use `roles.guard.ts` + `roles.decorator.ts` to restrict publishing and account linking.

Phase 8 — Observability and SLOs

- **Tracing**:
  - Spans: `variation.generate`, `media.prepare`, `platform.publish.*`.
- **Dashboards**:
  - Publish throughput, success rate, avg time-to-publish, error codes by platform, rate-limit events.
- **Alerts**:
  - Consecutive failures > N, rate-limit saturation, token expiry within 7 days.

Phase 9 — Rollout and Testing

- **Sandboxes**:
  - Use test org/page where supported; X staging keys if available.
- **E2E tests**:
  - Mock adapters in CI; record/replay (VCR-style) for non-deterministic APIs.
- **Feature flags**:
  - Incremental: enable per-channel and per-account; start with dry-run (log-only), then limited real posts.

Concrete Deliverables

- Endpoints: `POST /content/:id/publish`, `GET /content/:id/publish-status`, `DELETE /publish/:id`.
- Queues: `content.adapt`, `content.publish`, `content.publish.retry`, `metrics.fetch`.

- Services: channel adapters completed, scheduler wired, status tracker, metrics ingestor.
- Docs: runbook for token renewal, rate-limit remediation, and rollback; Postman updates; dashboards.

Day-by-Day (2 Weeks)

- **Day 1–2**: OAuth flows + `ConnectedChannel`; scopes verified; token refresh tested.
- **Day 3–4**: Channel adapters w/ publish and delete; per-channel validators.
- **Day 5**: Scheduler + `content.publish` worker; idempotency; dry-run.
- **Day 6**: `POST /content/:id/publish` + status endpoint; UI surfaces (optional).
- **Day 7**: Webhooks/polling ingest; store `platformPostId`, permalinks.
- **Day 8**: Retry/backoff, error taxonomy, partial failure UX.
- **Day 9**: Metrics fetchers + dashboards; alerts.
- **Day 10**: Security/RBAC, audit logs; feature flags; E2E tests and canary rollout.
- In short: generate → validate/approve → adapt per channel → schedule → post via adapters → track → retry/rollback → measure and learn.

(Non-Technical) Roadmap

Goal

Align the Content Engine to reliably create, approve, and auto-publish channel-ready posts that drive engagement and leads, with minimal manual effort and clear oversight.

What “Good” Looks Like

- Content briefs turn into ready-to-publish posts across LinkedIn, Instagram, Facebook, and X.

- Brand tone, SEO, and compliance are consistent.
- Approvals and scheduling are simple and reliable.
- Performance is measured; learnings loop back to improve future content.

Key Stakeholders

- Marketing: briefs, approvals, brand voice, performance insights
- Sales/Success: lead quality, escalations from social
- Compliance/Legal: policy checks, audit trails
- Ops/IT: reliability, access control, security
- Leadership: ROI, reach, leads, velocity

Core Journeys (Non-Technical)

1. Brief to Draft:
    - Marketing enters a topic, target audience, and desired outcome.
    - Engine proposes outline and first draft with sources.
2. Optimize & Approve:
    - Engine suggests SEO keywords, titles, and meta.
    - Brand/Compliance review and approve or request edits.
3. Adapt & Schedule:
    - Engine creates tailored versions for each social channel.
    - Team selects accounts, sets schedule, and final CTAs.
4. Auto-Publish & Track:
    - Posts go live automatically at the chosen times.
    - Engagement and reach are tracked on each channel.
5. Learn & Improve:
    - Reports surface what worked.
    - Templates and prompts are refined accordingly.

Non-Technical Roadmap (8 Steps)

1. Align Strategy and Guardrails

- Define brand voice, tone ladders, and no-go lists.
- Set target KPIs (reach, CTR, leads, time-to-publish).
- Decide channels, posting cadence, and “must-have” content formats.

1. Prepare the Content System

- Create a simple template library (blog, newsletter, LinkedIn, IG, X, FB).
- Standardize briefs: objective, audience, offer, CTA, references.
- Establish an approval policy (who approves, SLA, exceptions).

1. Build the Editorial Calendar

- Create a rolling 4–6 week calendar by theme/campaign.
- Slot posts per channel and audience segment.
- Mark “anchor” content (blogs/newsletters) and derivative social posts.

1. Set Up Accounts and Governance

- Centralize social account access under the brand.
- Assign roles: Content Creator, Approver, Publisher, Analyst.
- Document data handling rules (PII, screenshots, claims, disclosures).

1. Content Lifecycle (Day-to-Day)

- Intake: Marketing submits a brief for each anchor topic.
- Draft: Engine creates draft + suggested titles, hooks, and CTAs.
- Optimize: SEO suggestions + readability + link suggestions.
- Validate: Brand safety, plagiarism, sentiment, claims confidence.
- Approve: One-click approve or request changes.
- Adapt: Auto-generate channel-specific versions with hooks/hashtags.
- Schedule: Pick times per channel; add tracking (UTMs).
- Publish: Auto-post; optional “first comment” for hashtags/links.

- Monitor: Daily glance at engagement; triage escalations.

1. Engagement and Lead Handling

- Response playbook by intent (interest, demo, support, complaint).
- Escalation rules: high-risk terms, negative sentiment, VIPs.
- Lead capture: collect contact details from inbound comments/DMs.
- Sales handoff: notify with context and confidence level.

1. Measurement and Feedback Loop

- Weekly scorecard:
  - Reach, engagement rate, CTR, saves/shares, leads, sentiment trend.
- Content learning:
  - Which hooks, formats, times, and channels perform best.
- Template evolution:
  - Update prompts and channel guidelines based on winners.

1. Risk Management and Change Control

- Review cycle for policy/legal-sensitive posts.
- Fail-safe: pause all scheduled content during incidents.
- Incident playbook: mispost, negative virality, or account lockouts.
- Quarterly audit: access, tokens, content policy compliance.

Phased Timeline (5 Weeks)

- Week 1: Strategy & Foundations
  - Finalize voice, KPIs, approval chain, and calendars.
  - Inventory accounts and assign roles.
- Week 2: Content Workflows
  - Create templates and brief forms.
  - Pilot 2–3 anchor topics into drafts and approvals.

- Week 3: Channel Adaptation & Scheduling
    - Produce platform-ready variations from approved content.
    - Dry-run scheduling (no posting) to validate timing and formatting.
- Week 4: Auto-Publish & Engagement
    - Turn on automated posting with canary accounts.
    - Implement response playbook and escalation routines.
- Week 5: Optimize & Scale
    - Review metrics; refine prompts and templates.
    - Expand to more brands/regions; raise posting volume.

Success Metrics

- Content velocity: time from brief to approved post.
- Consistency: brand/SEO checklist pass rate.
- Channel performance: engagement rate, CTR, saves/shares.
- Business impact: qualified leads and pipeline touches.
- Operational reliability: on-time publishing, low error rate.

Policies and Checklists

- Brand checklist: tone, CTA, visuals, claims.
- SEO checklist: title, keywords, internal links, readability.
- Social checklist: platform limits, hashtags, mentions, alt text.
- Compliance checklist: disclosures, restricted topics, PII redaction.

Change Management and Training

- Short enablement sessions for Creators, Approvers, and Analysts.
- One-page quick-start guides for briefs, approvals, scheduling.
- Monthly office hours for template and playbook evolution.

Rollout Plan

- Start with a single brand and two channels (e.g., LinkedIn + Instagram).
- Enable auto-posting for approved content only.
- After 2 weeks of clean performance, expand channels and posting frequency.
- Establish quarterly reviews to refresh goals and guardrails.

If you want, I can tailor this roadmap to your current campaigns and build a 4–6 week editorial calendar with themes, offers, and posting slots.

No-touch content engine

Short answer: Yes—“no-touch” is feasible, but only if we put strong guardrails, clear inputs, and automated fail-safes in place. Think of it as “hands-off by default, human-in-loop only on exceptions.”

What you must lock down first

- **Brand guardrails**: voice/tone, do/don’t lists, compliance rules, approved CTAs, claim policies.
- **Templates**: briefs, prompts, and per-channel post templates that already reflect brand/SEO rules.
- **Approval policy**: auto-approve if quality/confidence ≥ thresholds; auto-hold if below.

How a no-touch flow would work (end-to-end)

1. Intake

- Content themes sourced from your calendar and campaign plan.
- Auto-generate briefs from themes or ingest from a sheet—no manual form-filling.

1. Draft and Optimize

- Engine creates draft + outline + sources.
- Auto-SEO: keywords, titles, meta; readability fixes applied automatically.

1. Quality Gate

- Automated safety, plagiarism, factuality, style checks.
- Auto-approve if scores meet thresholds; auto-revise once; if still low, auto-hold and alert.

1. Channel Adaptation

- Platform-ready variations (length, hashtags, handles, alt text) created automatically.
- A/B variants for hooks/CTAs generated and scored; best variant auto-selected.

1. Scheduling and Posting

- Uses your posting cadence and audience timezones.
- Auto-scheduling within send windows; posts published with tracking (UTMs).

1. Monitoring and Engagement

- Auto-ingest comments/mentions; rules engine replies to common intents.
- Negative sentiment, VIP, or compliance risk auto-escalates to human.

1. Learning Loop

- Weekly auto-report of winners (hooks, formats, times).
- Templates and prompts auto-tuned within safe bounds; larger changes gated.

1. Governance and Safety

- Global kill-switch to pause all scheduled posts.
- Token/permission health checks; audit logs; rate-limit and rollback policies.

Where you still need humans (minimal)

- Policy/claims updates and crisis comms.
- Edge cases that fail confidence or compliance thresholds.

- Quarterly review of templates, tone, and KPIs.

Maturity levels to reach “no-touch”

- Level 1: Draft to approve (manual final check).
- Level 2: Auto-approve on high confidence; scheduled auto-post.
- Level 3: Auto-engage replies for routine intents; escalate exceptions.
- Level 4: Continuous auto-optimization (A/B learnings) with guardrails.

Configuration to enable “no-touch”

- Thresholds: safety ≥ 0.9, plagiarism ≤ 0.15 overlap, factuality ≥ 0.8, style/readability within target.
- Channels: start with 1–2 (e.g., LinkedIn + Instagram) before expanding.
- Cadence: daily/weekly slots per channel; quiet hours; launch windows.
- Escalation: define VIPs, blocked terms, legal triggers, and SLA routes.

Risks and mitigations

- Policy drift → quarterly guardrail review and template versioning.
- Platform changes → channel rules stored in config; monitored alerts.
- Reputational risk → kill-switch + auto-hold on low confidence or trend-negative sentiment.

If you want, I can map your current campaigns and propose a 4–6 week, truly “no-touch” calendar with pre-set thresholds and escalation paths so you can turn it on safely.