Creator Autopilot MVP (Instagram Reels + YouTube)

1) Team Split (2 engineers, equal load)

Engineer A — Platform Backend + Data + Auth

Owns the product backend that the web app and n8n talk to.

Responsibilities

1. Auth + workspace
2. DB schema + migrations
3. Core REST API (content, templates, scheduling, posts, analytics)
4. Media storage service (S3/R2/GCS) + **public URL generation** for IG
5. Webhook endpoints for n8n callbacks + job state machine
6. Observability: logs, errors, audit trail

Deliverables

- Backend service running with DB + migrations
- API docs (OpenAPI/Swagger)
- Storage pipeline working (upload → asset record → public URL)
- Clean status transitions: `planned → generating → needs_review → approved → scheduled → publishing → posted/failed`
- Webhook receivers for n8n: `/webhooks/n8n/...`

Engineer B — Integrations + n8n Orchestration + Platform APIs

Owns execution via n8n and all external API integrations.

Responsibilities

1. n8n instance setup (secure, env vars, secrets)
2. Token usage strategy (how n8n obtains tokens safely)
3. Meta Reels publish workflow (container → publish)
4. YouTube resumable upload workflow
5. Analytics sync workflows (IG + YT metrics pull)
6. Generation workflow skeleton (planner/judge/generator placeholders first)
7. Retry + idempotency strategy inside workflows

Deliverables

- n8n workflows: `content.publish`, `analytics.sync`, `content.generate` (stub → real)
- Working publishing to:
  - Instagram Reels (Meta Graph API)
  - YouTube (resumable upload)
- Callbacks to backend with results + errors

---

2) Backend-First Build Plan (rapid + organized)

Phase 0 — Contract First (same day)

Both engineers agree before coding

A) Status Machine (single source of truth)

- `planned`
- `generating`
- `needs_review`
- `approved`
- `scheduled`
- `publishing`
- `posted`
- `failed`

B) Shared JSON Shapes

- `plan_json` schema (planner output)
- `judge_json` schema (judge output)
- `metrics_json` schema (analytics snapshot)

C) Required Webhooks (n8n → backend)

- `POST /webhooks/n8n/content-generated`
- `POST /webhooks/n8n/content-published`
- `POST /webhooks/n8n/analytics-synced`

Goal: lock contracts early to avoid rewrites.

---

3) Phase 1 — Foundations (Days 1–2)

Engineer A (Days 1–2)

1. Repo + project setup (TS backend)
2. DB setup (Postgres recommended)
3. Migrations tool (Prisma/Drizzle/Knex)
4. Create MVP tables:
   - `users`, `workspaces`
   - `connected_accounts`
   - `templates`, `brand_profiles`
   - `content_items`, `assets`, `reviews`
   - `schedule`, `published_posts`, `analytics_snapshots`
5. Auth endpoints:
   - `POST /auth/signup`, `POST /auth/login`
6. Storage service:
   - `POST /assets/presign` (signed upload URL)
   - `POST /assets/complete` (register asset)
   - Ensure **public CDN URL** generation for Instagram `video_url`

End of Day 2: backend running + auth + DB + assets.

Engineer B (Days 1–2)

1. Deploy n8n (Docker)
2. Security hardening:
   - Basic auth
   - Restrict editor access
   - Secure credentials
3. Create credential placeholders (Meta/Google)
4. Verify backend ↔ n8n connectivity:
   - n8n webhook → backend `/webhooks/n8n/ping`
   - backend trigger endpoint → n8n

End of Day 2: backend ↔ n8n communication verified.

---

4) Phase 2 — Core APIs + Publish Workflows (Days 3–5)

Engineer A (Days 3–5)

Content APIs

- `POST /content` (create item(s))
- `GET /content`
- `GET /content/:id`
- `POST /content/:id/approve`

- `POST /content/:id/reject`
- `POST /content/:id/regenerate`

Template APIs

- `GET /templates` (system + user)
- `POST /templates` (save)
- `PATCH /templates/:id`

Scheduling APIs

- `POST /schedule`
- `GET /schedule`
- `PATCH /schedule/:id`

Webhook receivers (n8n → backend)

- `POST /webhooks/n8n/content-generated`
- `POST /webhooks/n8n/content-published`
- `POST /webhooks/n8n/analytics-synced`

Idempotency (required): accept duplicate callbacks safely using:

- `(content_item_id + event_type + event_id)` unique key

Engineer B (Days 3–5)

Instagram Reels publish workflow (n8n)

Workflow: `content.publish.instagram`

Input:

- `content_item_id`
- `ig_user_id`
- `access_token`
- `video_url`
- `caption`

Steps:

1. Create container: `POST /{ig_user_id}/media?media_type=REELS&video_url=...&caption=...`
2. Publish: `POST /{ig_user_id}/media_publish?creation_id=...`
3. Callback backend `content-published` with:

    - `platform_post_id`, `platform_url`, `status`

YouTube publish workflow (n8n)

Workflow: `content.publish.youtube`

Input:

- `content_item_id`
- `access_token/refresh_token`
- `video_url`
- `title`, `description`, `tags`

Steps:

1. Fetch/stream video
2. Resumable upload to YouTube (`videos.insert` via resumable protocol)
3. Callback backend `content-published` with:
    - `videoId`, `platform_url`, `status`

Unified publish workflow

Workflow: `content.publish`

- Routes IG/YT/both
- Retry policy + failure callback

End of Day 5: posting works end-to-end **with manually provided tokens**.

---

5) Phase 3 — Account Connections (OAuth) + Token Storage (Days 6–8)

Engineer A (Days 6–8)

Implement connected account endpoints + token storage:

- `GET /integrations`
- `POST /integrations/youtube/connect` (OAuth start)
- `GET /integrations/youtube/callback`
- `POST /integrations/instagram/connect`
- `GET /integrations/instagram/callback`
- Token refresh logic (cron or on-demand)

Store in `connected_accounts`:

- `access_token`, `refresh_token`, `expires_at`, `scopes`
- `platform_user_id` / `ig_user_id` / `youtube_channel_id`

Engineer B (Days 6–8)

Update n8n workflows to **fetch tokens from backend** (avoid passing tokens from UI):

1. Workflow starts with `content_item_id`
2. n8n calls backend: `GET /internal/content/:id/publish-payload`
3. Backend returns required tokens + IDs + caption/title
4. n8n publishes
5. n8n callbacks update DB

End of Day 8: real OAuth connections; no manual tokens.

---

6) Phase 4 — Generation Pipeline Skeleton (Days 9–10)

Engineer A (Days 9–10)

Ship “publish from uploaded video” mode to unblock product:

- `POST /content/from-upload`
- `POST /content/:id/set-caption`
- Ensure asset linking + preview URL

Engineer B (Days 9–10)

Create `content.generate` workflow skeleton (placeholders first):

- Planner agent (mock JSON)
- Generator placeholder (reuses uploaded video or simple render)
- Judge placeholder (pass/fail)
- Callback backend `content-generated` with:
    - `plan_json`, `judge_json`, `assets[]`

---

7) Phase 5 — Analytics + Dashboard Data (Days 11–12)

Engineer A (Days 11–12)

Analytics endpoints + aggregation:

- `GET /analytics/summary`

- `GET /analytics/posts/:id`
- Basic rollups by platform/template

Engineer B (Days 11–12)

Workflow: `analytics.sync`

1. Fetch posts needing refresh from backend
2. Pull metrics from IG + YT APIs
3. Callback backend `analytics-synced`
4. (Optional v1) Insights agent to produce recommendations

---

8) Daily Execution Routine (to finish rapidly)

Standup (10 minutes)

1. What endpoint/workflow shipped yesterday?
2. Any contract changes (payload/schema)?
3. What is blocked?

Anti-blocker rule

- If Engineer B needs an endpoint, Engineer A ships a **stub** returning mock payload same day.
- If Engineer A needs publish results, Engineer B provides a **mock callback** payload same day.

---

9) Milestones (clear checkpoints)

Milestone 1 (End Day 5)

✅ Upload video manually → schedule → n8n publishes to IG/YT → backend shows `posted`.

Milestone 2 (End Day 8)

✅ OAuth connections stored → publish without manual tokens.

Milestone 3 (End Day 12)

✅ Analytics sync + dashboard endpoints.

Milestone 4 (Week 3+)

✅ Full studio: planner → generator → judge → approve → schedule → post.

---

10) Immediate Next Actions (today)

Engineer A starts

1. DB schema + migrations
2. Auth endpoints
3. Storage presign endpoints
4. Minimal Content CRUD endpoints

Engineer B starts

1. n8n setup + secure
2. `content.publish.instagram` workflow
3. Backend webhook callback integration test