# Teoram CMS — Article Pipeline (Canvas Overview)

This document captures **Phase 3 — Article Pipeline** in canvas format, aligned with the Prisma schema and tables we already have (Topic, Source, Article, Citation).

---

## 🎯Goal

Transform discovered topics into **high-quality, research-backed articles** with clear review, approval, and publishing steps — all observable in CMS.

---

## 📷Status Machine (per Topic)

- **NEW** → Discovered & grouped, awaiting curator.
- **APPROVED** → Curator approves sources.
- **PROCESSING** → Agent job started.
- **COLLECTED** → Sources scraped, transcripts pulled.
- **DRAFTED** → Gemini draft ready.
- **ASSIGNED** → Category/subcategory + type assigned.
- **READY** → QA passed, ready to publish.
- **PUBLISHED** → Live.

---

## 🎙️ Pipeline Steps

### 1. Discovery (already done)

- Fetch tech-only trends (Google Trends + YouTube).
- Group with Gemini → master topics.
- Canonicalize with Qdrant.
- Save to Postgres (Topics + Sources).

### 2. Approval (Human-in-loop)

- Curator selects sources in CMS.
- Approves → Topic status → `APPROVED`.
- Prisma updates: `Topic.status`, `Source.approved`.

### 3. Collect Content (Agent)

- Crawl approved sources → extract HTML/transcripts.
- Normalize into `blocks[]` with citations.
- Save as **Citation** rows linked to Article.

- Status: `COLLECTED`.

### 4. Draft Article (Agent)

- Input: Topic + Sources + Citations.
- Gemini generates `{ title, tl_dr, body_html, faq_html, outline_json, metaTitle, metaDescription }`.
- Prisma `Article` created.
- Status: `DRAFTED`.

### 5. Review / QA

- Automated LLM QA: clarity, citations, style.
- Human editor review in CMS.
- Status: `READY` or `NEEDS_CHANGES`.

### 6. Categorize & Publish

- Assign category/subcategory (auto + human override).
- Prisma `Article.publishedAt` set.
- Status: `PUBLISHED`.

### 7. Post-Publish — Sentiment & Scores

- Fetch Reddit, YouTube comments, X/Twitter, etc.
- Sentiment analysis + product scoring.
- Store on `Article` or related `Product` entity.
- Exposed in CMS widgets.

---

## Data Shapes (Prisma Models)

### Topic

```
model Topic {
  id        String   @id @default(uuid())
  slug      String   @unique
  title     String
  status    TopicStatus
  score     Float?
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt
  sources   Source[]
  articles  Article[]
}
```

### Source

```
model Source {
  id        String   @id @default(uuid())
  url       String
  kind      SourceKind
  topicId   String
  approved  Boolean  @default(false)
  createdAt DateTime @default(now())
  topic     Topic    @relation(fields: [topicId], references: [id])
}
```

### Article

```
model Article {
  id             String   @id @default(uuid())
  topicId        String
  sourceId       String?
  title          String
  tl_dr          String?
  body_html      String
  faq_html       String?
  outline_json   Json?
  metaTitle      String?
  metaDescription String?
  publishedAt    DateTime?
  createdAt      DateTime @default(now())
  topic          Topic    @relation(fields: [topicId], references: [id])
  citations      Citation[]
}
```

### Citation

```
model Citation {
  id         String   @id @default(uuid())
  articleId  String
  sourceUrl  String
  sourceType String
  title      String?
  author     String?
  timestamp  DateTime?
  quote      String?
  article    Article  @relation(fields: [articleId], references: [id])
}
```

---

## 🏓 CMS Views (Next.js)

- **Dashboard**: Stats + trigger discovery.
- **Discovery Board**: Master topics, approve sources.
- **Topic Detail**: Step tracker (NEW → PUBLISHED).
- **Article Draft View**: Preview body, approve QA.
- **Published Articles**: With sentiment & product scores.

---

## Acceptance Checklist

- [ ] Deduplicated topics & sources.
- [ ] Qdrant canonicalization reused.
- [ ] Gemini drafts contain citations.
- [ ] QA pipeline enforces clarity & safety.
- [ ] Prisma writes align with schema.
- [ ] Publishing revalidates ISR pages.
- [ ] Metrics & logs track per-step progress.