WebRend AI Blog: Cron + Generation Overview

This document summarizes how the automated AI blog pipeline works in WebRend, including the daily cron, scheduler, news scraping, OpenAI generation, and how articles appear on the site.

System Components

- scripts/cron.js — boots a daily job at 8:00 AM PT and triggers the scheduler
- scripts/scheduler.ts — orchestrates guarded runs (lock file, last-run check), supports once/daemon
- scripts/news-scraper.ts — fetches news (NewsData, NewsAPI, GNews), calls OpenAI, saves to Firestore
- app/blog/page.tsx — lists recent posts from Firestore
- app/blog/[slug]/BlogSlugPage.tsx — renders a full article

How It Runs Daily

scripts/cron.js creates a log directory, writes a startup flag, and decides whether to run immediately if this is the first boot or if no run has happened today. It then schedules a CronJob at 08:00 AM America/Los_Angeles. The cron executes the scheduler with ts-node ESM.

```
npm run cron
```

Scheduling Modes

scripts/scheduler.ts loads env (.env.local), enforces a lock file (article-generation.lock) to prevent overlapping runs, and tracks last run in article-generation-last-run.json. It supports:

- once (default) — run a single generation cycle and exit
- daemon — run immediately then schedule the next daily 8:00 AM run via timers

```
npm run generate-articles
# or
npm run start-article-generation
```

Generation Pipeline

scripts/news-scraper.ts tries NewsData.io !’ NewsAPI !’ GNews. For each fetched item (limited to 2 per run), it:

- Checks Firestore for duplicates (sourceUrl and title similarity)

  - Classifies a category via OpenAI (gpt-4o-mini)
- Generates 1000–1500 word Markdown content via OpenAI
- Computes read time, builds a slug, selects a safe Unsplash image URL
- Saves the article into Firestore (collection: articles)

```
env required: OPENAI_API_KEY, FIREBASE_* (PROJECT_ID, CLIENT_EMAIL,
PRIVATE_KEY), NEWSDATA_API_KEY or NEWS_API_KEY or GNEWS_API_KEY
```

Displaying Articles

app/blog/page.tsx queries Firestore (ordered by publishedAt desc) and passes articles to BlogPageClient. Each article needs title, description, publishedAt, imageUrl, category, readTime, slug.

The slug page component app/blog/[slug]/BlogSlugPage.tsx renders Markdown, header meta, image, source link, and share buttons.

Operations

- Logs are written under logs/ (cron-YYYY-MM-DD.log, article-generation-YYYY-MM-DD.log)
- Initial run guard: .cron-initial-run-completed file prevents duplicate startup executions
- Locking: article-generation.lock avoids concurrent runs; stale (>2h) locks are overwritten
- Last-run file: article-generation-last-run.json blocks same-day repeats unless forced
- You can create a manual test post with scripts/create-test-article.js

```
node --loader ts-node/esm scripts/create-test-article.js
```

Commands Quick Ref

```
npm run cron
npm run generate-articles
npm run start-article-generation
node --loader ts-node/esm scripts/news-scraper.ts
```

Troubleshooting

- Missing env keys !’ generation will fail early; check .env.local
- Firestore index warnings are printed with a console URL to create the index
- Image URL sanitization may reject non-direct or untrusted image hosts
- If no news provider returns results, the run aborts with an error log

For details, review the source files mentioned above.