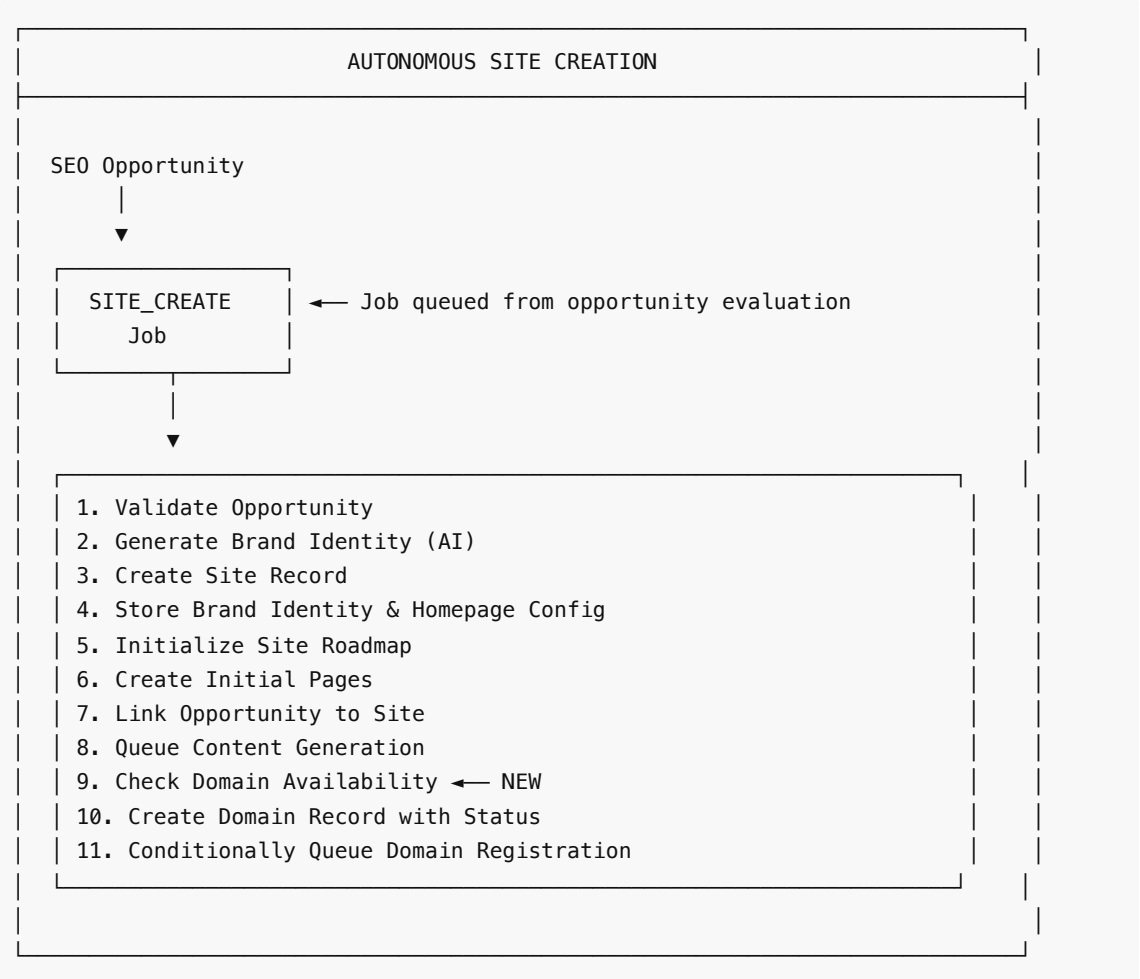
Autonomous Site Creation Flow

This document describes the complete autonomous flow for creating micro-sites from SEO opportunities.

Flow Overview



Detailed Step-by-Step Flow

Step 1: Validate Opportunity

File: `packages/jobs/src/workers/site.ts:53-69`

- Fetches the SEO opportunity from database
- Validates opportunity exists and isn't already assigned
- Checks opportunity status is `IDENTIFIED`, `EVALUATED`, or `ASSIGNED`

Step 2: Generate Brand Identity

File: `packages/jobs/src/workers/site.ts:71-92`

Uses AI to generate comprehensive brand identity:

- Brand name and tagline

- Color palette (primary, secondary, accent)
- Typography (heading/body fonts)
- Tone of voice and personality
- Content guidelines and semantic keywords

Step 3: Create Site Record

File: `packages/jobs/src/workers/site.ts:96-130`

- Generates URL slug from brand name
- Checks for slug collisions
- Creates site with `DRAFT` status
- Creates associated brand record

Step 4: Store Brand Identity & Homepage Config

File: `packages/jobs/src/workers/site.ts:132-152`

- Stores extended brand identity in `seoConfig`
- Generates AI-powered homepage configuration:
  - Hero section content
  - Popular experiences settings
  - Destination suggestions
  - Testimonials

Step 5: Initialize Site Roadmap

File: `packages/jobs/src/workers/site.ts:154-155`

Creates planned tasks for the site including:

- Content generation tasks
- SEO optimization tasks
- GSC verification tasks

Step 6: Create Initial Pages

File: `packages/jobs/src/workers/site.ts:157-159`

Creates page records for:

- Homepage
- About Us
- Contact
- Privacy Policy
- Terms of Service

Step 7: Link Opportunity to Site

File: `packages/jobs/src/workers/site.ts:161-170`

- Updates opportunity with `siteId`
- Sets opportunity status to `ASSIGNED`

Step 8: Queue Content Generation

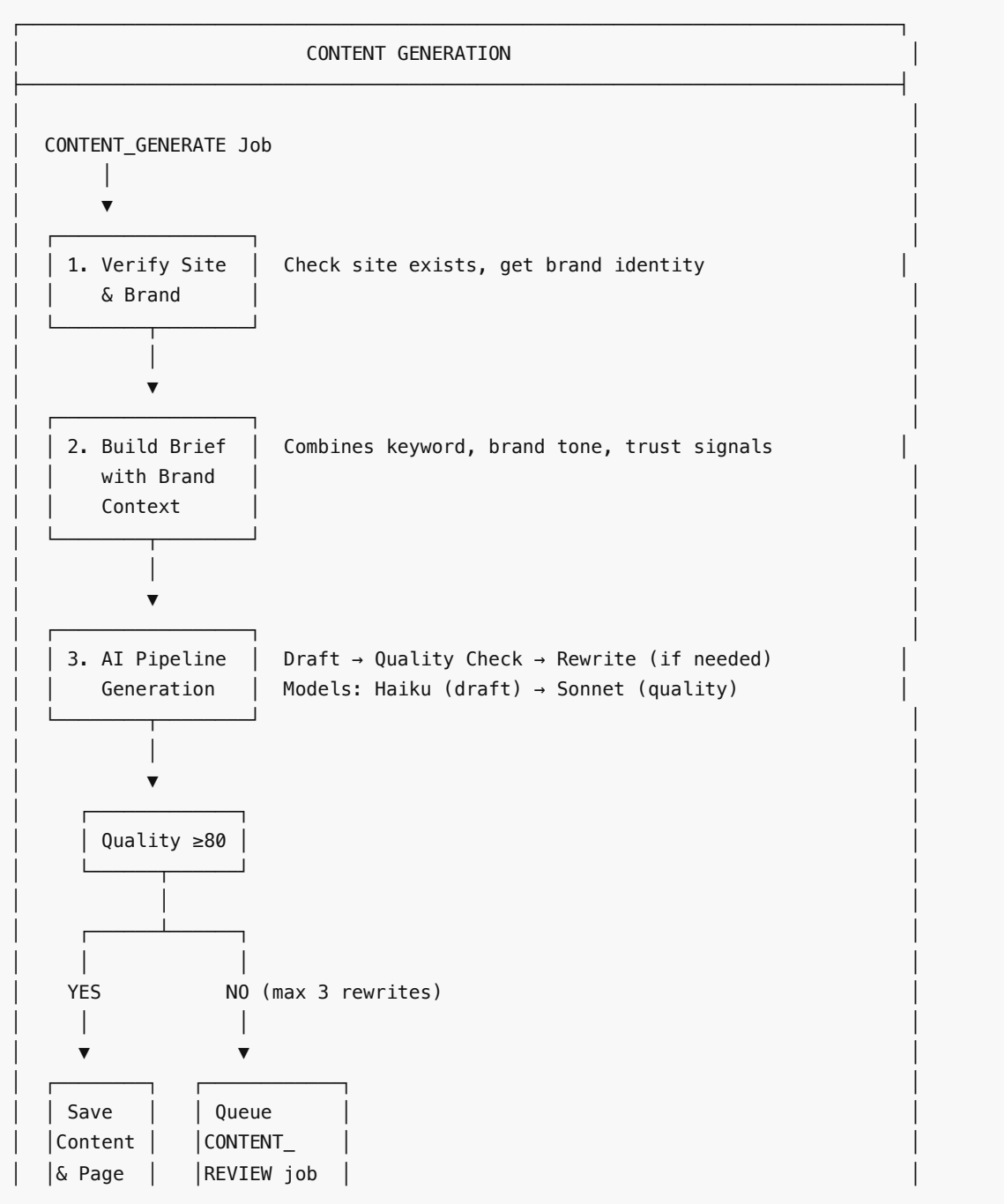
File: `packages/jobs/src/workers/site.ts:172-185`

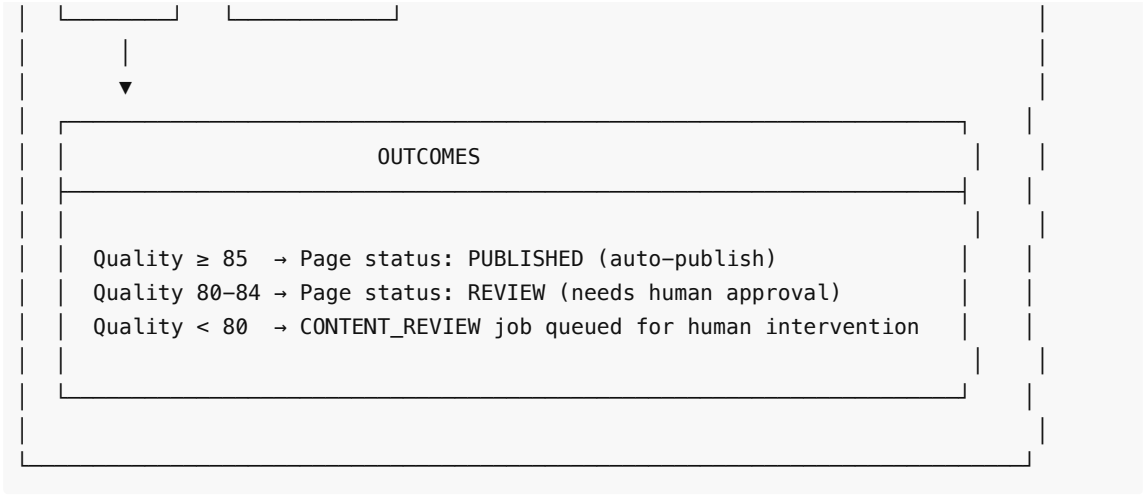
Queues `CONTENT_GENERATE` job with:

- Site ID
- Opportunity ID
- Content type (`destination`)
- Target keyword from opportunity
- Secondary keywords from brand identity semantic keywords

Content Generation Flow

When `CONTENT_GENERATE` is queued, it triggers the AI content pipeline:





Content Generation Details

File: `packages/jobs/src/workers/content.ts:26–249`

Step 1: Verify Site & Get Brand Identity

- Fetches site from database
- Retrieves comprehensive brand identity including:
  - Tone of voice (personality, writing style)
  - Trust signals (value propositions, social proof)
  - Brand story (mission, origin)
  - Content guidelines (semantic keywords)

Step 2: Build Content Brief

```
brief = {
  type: contentType,
  siteId,
  siteName: site.name,
  targetKeyword,
  secondaryKeywords,
  destination: opportunity?.location,
  category: opportunity?.niche,
  targetLength: { min: 800, max: 1500 },
  brandContext: {
    toneOfVoice,
    trustSignals,
    brandStory,
    contentGuidelines,
    writingGuidelines,
  }
}
```

Step 3: AI Pipeline Generation

- Uses content-engine pipeline with circuit breaker protection
- Draft model: Haiku (fast, cost-effective)

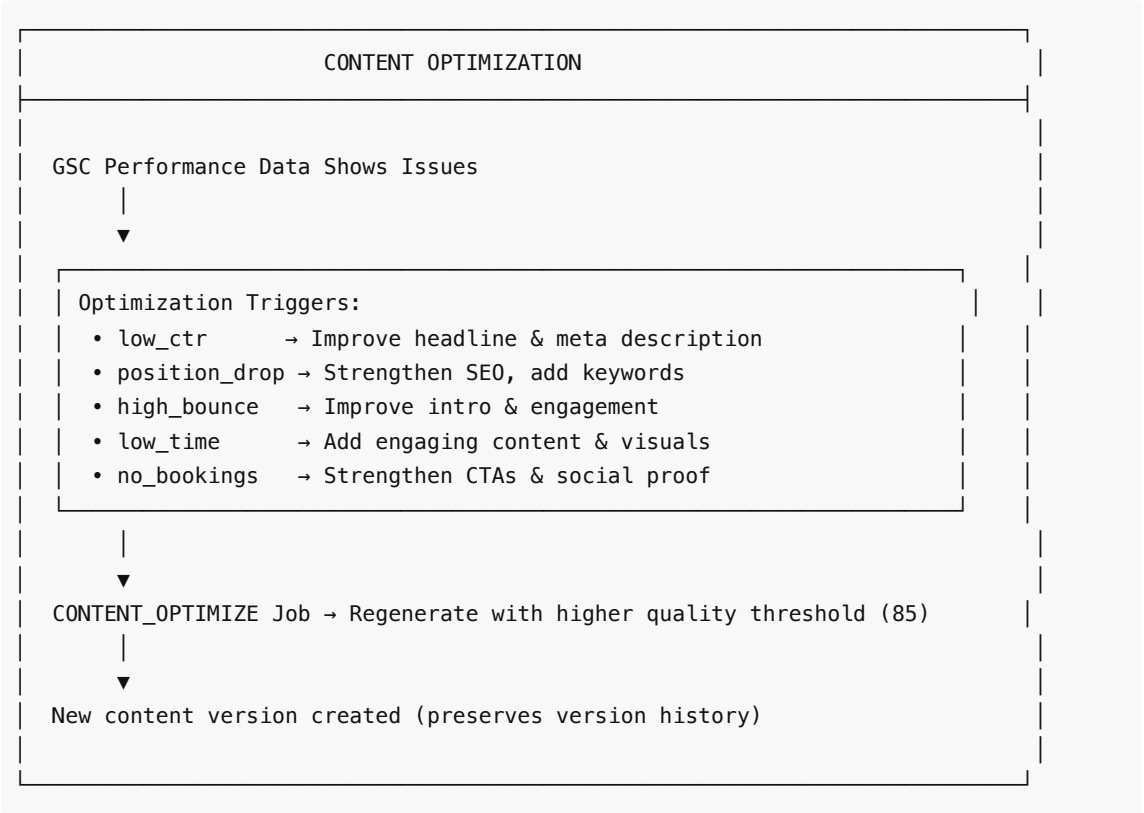
- Quality assessment model: Sonnet (more capable)
- Rewrite model: Haiku (up to 3 attempts)
- Quality threshold: 80/100

Step 4: Save Content & Create Page

- Creates `Content` record with:
  - Generated markdown body
  - AI model used
  - Quality score
  - Version number
- Creates or updates `Page` record
- Auto-publishes if quality score ≥ 85

Content Optimization Flow (Performance-Based)

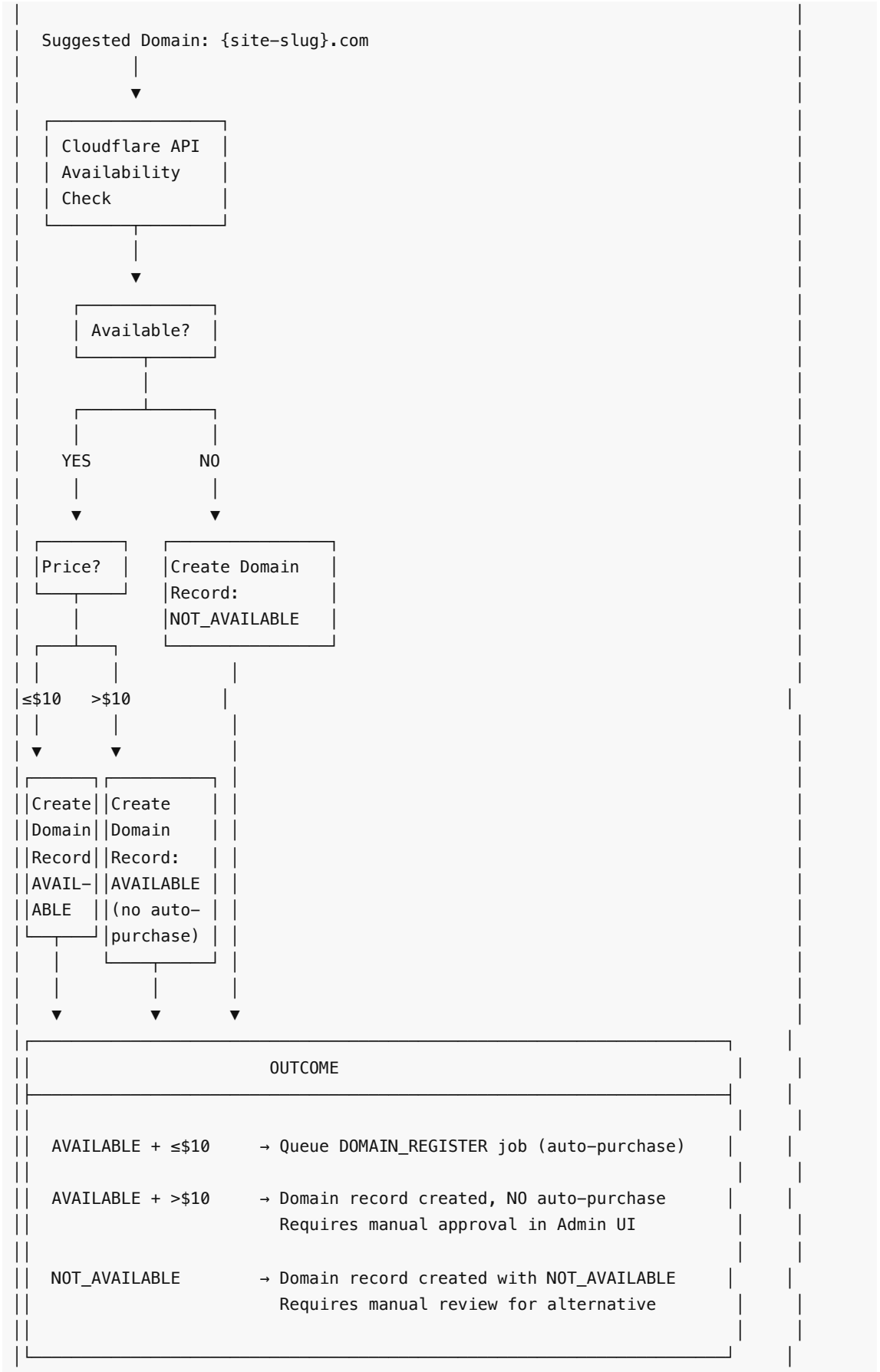
The system also includes automatic content optimization based on GSC performance data:



File: `packages/jobs/src/workers/content.ts:255-422`

Domain Availability Flow (Steps 9-11)

This is the newly integrated availability checking that happens during site creation.


Suggested Domain: {site-slug}.com
Cloudflare API Availability Check
Available?
YES
NO
Price?
Create Domain Record: NOT_AVAILABLE
≤$10
>$10
Create Domain Record AVAIL-ABLE
Create Domain Record: AVAILABLE (no auto-purchase)
OUTCOME
AVAILABLE + ≤$10 → Queue DOMAIN_REGISTER job (auto-purchase)
AVAILABLE + >$10 → Domain record created, NO auto-purchase
Requires manual approval in Admin UI
NOT_AVAILABLE → Domain record created with NOT_AVAILABLE
Requires manual review for alternative

```
│                                                                              │
└──────────────────────────────────────────────────────────────────────────────┘
```

Step 9: Check Domain Availability

File: `packages/jobs/src/workers/site.ts:187-192`

```
const suggestedDomain = domain || `${site.slug}.com`;
const availabilityResult = await checkDomainAvailabilityForSite(suggestedDomain);
```

Calls Cloudflare Registrar API to check:

- Is domain available for registration?
- What is the registration price?

Step 10: Create Domain Record

File: `packages/jobs/src/workers/site.ts:194-209`

Creates a domain record in the database with:

- `AVAILABLE` status if domain can be purchased
- `NOT_AVAILABLE` status if domain is taken
- Stores the registration price for admin visibility

Step 11: Conditional Domain Registration

File: `packages/jobs/src/workers/site.ts:211-233`

Decision Logic:

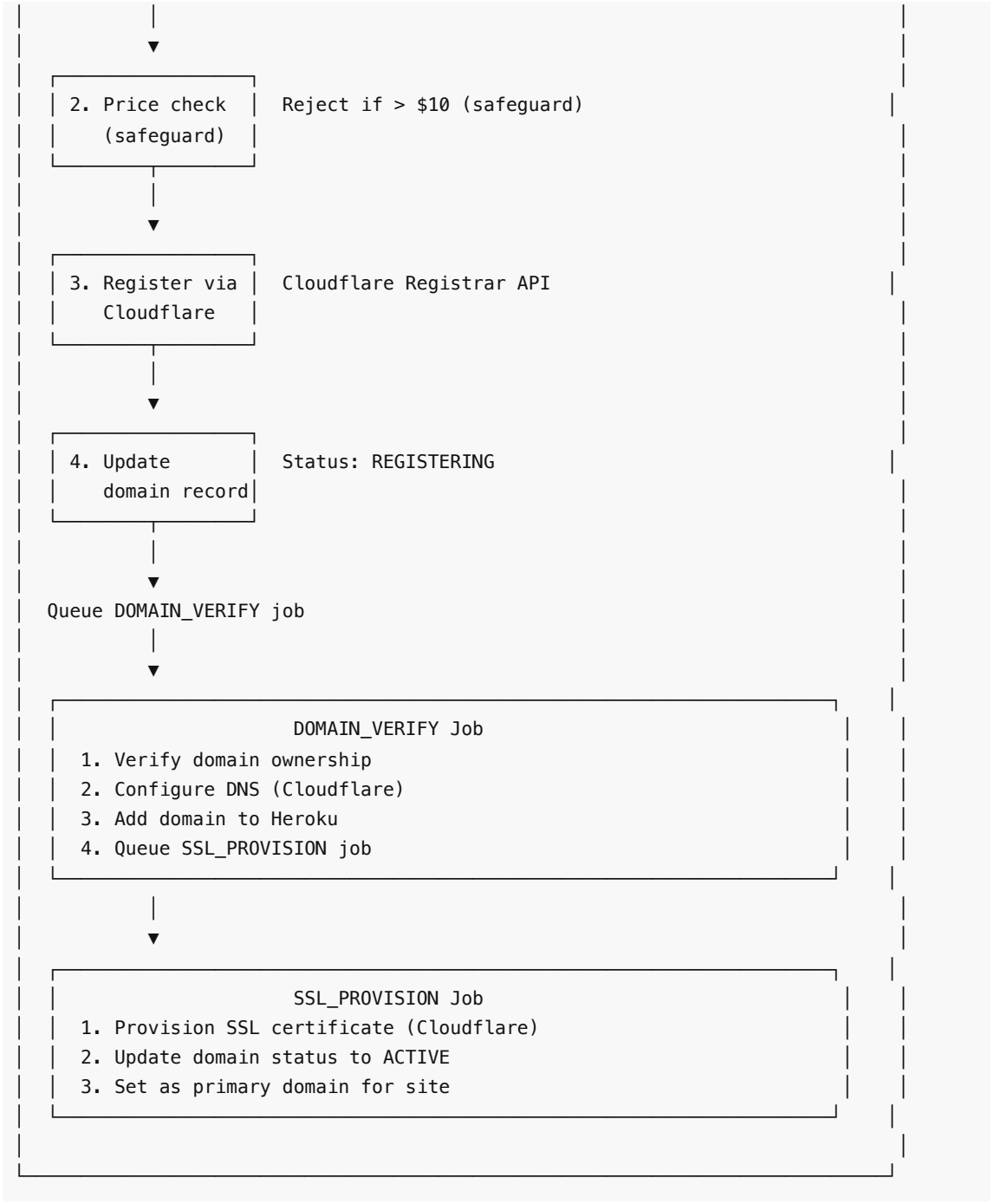
Condition	Action	Status
Available + Price ≤ $10	Queue `DOMAIN_REGISTER` job	Auto-purchase
Available + Price > $10	Log message, no job queued	Requires approval
Not Available	Log message, no job queued	Requires alternative

---

Domain Registration Flow (If Queued)

When `DOMAIN_REGISTER` is queued, it triggers a separate workflow:

```
┌──────────────────────────────────────────────────────────────────────────────┐
│                        DOMAIN REGISTRATION                                   │
├──────────────────────────────────────────────────────────────────────────────┤
│                                                                              │
│  DOMAIN_REGISTER Job                                                         │
│       │                                                                      │
│       ▼                                                                      │
│  ┌─────────────────┐                                                         │
│  │ 1. Re-verify    │                                                         │
│  │    availability │                                                         │
│  └────────┬────────┘                                                         │
```



Admin UI Visibility

The admin panel at `/admin/domains` shows all domains with their statuses:

Status	Description	Admin Action
`AVAILABLE`	Domain available for purchase	"Purchase Domain" button
`NOT_AVAILABLE`	Domain taken elsewhere	Shows "Domain taken" - needs alternative

`PENDING`	Legacy: not yet checked	"Check Availability" button
`REGISTERING`	Purchase in progress	Shows spinner
`DNS_PENDING`	Awaiting DNS propagation	Status indicator
`SSL_PENDING`	Awaiting SSL certificate	Status indicator
`ACTIVE`	Fully configured and live	"Visit Site" link
`FAILED`	Registration failed	Retry option

Price Visibility

For `AVAILABLE` domains, the UI shows:

- Estimated price per year
- Domains over $10 are available but weren't auto-purchased
- Operators can manually approve expensive domain purchases

Key Files Reference

File	Purpose
`packages/jobs/src/workers/site.ts`	Site creation handler
`packages/jobs/src/workers/content.ts`	Content generation/optimization handlers
`packages/jobs/src/workers/domain.ts`	Domain registration/verification handlers
`packages/jobs/src/services/brand-identity.ts`	Brand identity generation & storage
`packages/jobs/src/services/cloudflare-registrar.ts`	Cloudflare domain API
`packages/jobs/src/services/cloudflare-dns.ts`	Cloudflare DNS configuration
`packages/jobs/src/services/heroku-domains.ts`	Heroku domain configuration
`packages/content-engine/`	AI content pipeline (draft, quality, rewrite)
`apps/admin/src/app/domains/page.tsx`	Admin domains UI
`apps/admin/src/app/api/domains/route.ts`	Domains API endpoints

Environment Variables Required

```
# Cloudflare (domain registration & DNS)
CLOUDFLARE_API_KEY=
CLOUDFLARE_EMAIL=
CLOUDFLARE_ACCOUNT_ID=

# Heroku (hosting configuration)
HEROKU_API_KEY=
```

```
HEROKU_APP_NAME=

# AI (brand generation)
ANTHROPIC_API_KEY=

# Database
DATABASE_URL=

# Holibob (experience API)
HOLIBOB_PARTNER_ID=
```

Summary

The autonomous flow ensures:

1. **Sites are fully configured** - Brand, pages, content all generated automatically
2. **AI-powered content** - Uses Claude (Haiku/Sonnet) with brand tone of voice and quality gates
3. **Content quality control** - Auto-publish at 85+ quality, human review at 80-84, flagged below 80
4. **Performance optimization** - Automatic content rewrites based on GSC metrics (CTR, position, bounce)
5. **Domain costs are controlled** - Only domains ≤$10 are auto-purchased
6. **Visibility for operators** - All domain statuses visible in admin panel
7. **Manual override possible** - Expensive or unavailable domains can be handled manually
8. **Complete hosting setup** - DNS, SSL, and Heroku all configured automatically for purchased domains