Phase 7 Quality Assurance Framework - Developer Guide

Quick Start

1. Environment Setup

```
# Copy environment template
cp .env.example .env

# Edit .env with your values
DATABASE_URL=postgresql://...
NEXTAUTH_SECRET=your-secret-here
ENCRYPTION_KEY=your-encryption-key
```

2. Installation & Migration

```
# Install dependencies
npm install

# Generate Prisma client
npm run prisma:generate

# Check migration status
npx prisma migrate status

# Seed default rulebook
npm run tsx scripts/seed_rulebook.ts
```

3. Development Server

```
# Development mode
npm run dev

# Production build & test
npm run build
PORT=3001 npm start
```

API Endpoints

Global Rulebook API

Endpoint: `/api/rulebook`

GET - Retrieve Current Rulebook

```
curl -H "Authorization: Bearer YOUR_TOKEN" \
     http://localhost:3001/api/rulebook
```

Response:

```
{
  "id": "cldefault001",
  "version": 1,
  "rules": {
    "eeat": {
      "require_author_bio": true,
      "require_citations": true,
      "allowed_source_domains": ["*.gov", "*.edu"]
    },
    "enforcement": {
      "default_min_quality_score": 75,
      "tag_if_below": "review-needed"
    }
  },
  "sources": ["https://developers.google.com/..."],
  "updatedBy": "admin"
}
```

POST - Create New Rulebook Version

```
curl -X POST http://localhost:3001/api/rulebook \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  --data '{
    "rules": {
      "eeat": { "require_author_bio": true },
      "seo": { "title_length": {"min": 45, "max": 65} },
      "enforcement": { "default_min_quality_score": 80 }
    },
    "sources": ["https://example.com/guide"],
    "notes": "Updated quality threshold"
  }'
```

Site Strategy API

Endpoint: `/api/sites/[id]/strategy`

GET - Retrieve Site Strategy

```
curl -H "Authorization: Bearer YOUR_TOKEN" \
     http://localhost:3001/api/sites/SITE_ID/strategy
```

POST - Update Site Strategy

```
curl -X POST http://localhost:3001/api/sites/SITE_ID/strategy \
  -H "Authorization: Bearer YOUR_TOKEN" \
  -H "Content-Type: application/json" \
  --data '{
    "site_persona": "Technical authority in AI/ML",
    "target_audience": "Software developers and ML engineers",
    "eeat_guidelines": {
      "author_bio_template": "Expert AI researcher with 10+ years experience",
      "preferred_sources": ["arxiv.org", "papers.nips.cc"],
      "tone_of_voice": ["technical", "authoritative", "practical"]
    },
    "content_archetypes": [{
      "name": "Tutorial",
      "prompt_file": "tutorial_prompt.txt",
      "priority": 0.8
    }]
  }'
```

Authentication

Bearer Token Setup

1. **Create API Token** (one-time setup):

```
# Generate a secure token
node -e "console.log(require('crypto').randomBytes(32).toString('base64'))"

# Store in connections table via admin UI or SQL:
INSERT INTO connections (kind, "dataEnc") VALUES (
  'console_api_token',
  'BASE64_ENCRYPTED_TOKEN_DATA'
);
```

1. **Use Token in Requests**:

```
export BEARER_TOKEN="your_generated_token"
curl -H "Authorization: Bearer $BEARER_TOKEN" http://localhost:3001/api/rulebook
```

Rate Limits

- **Rulebook GET**: 10 requests/minute
- **Rulebook POST**: 5 requests/5 minutes
- **Strategy GET**: 20 requests/minute
- **Strategy POST**: 10 requests/5 minutes

Rate limit exceeded returns `429` with headers:

```
X-RateLimit-Limit: 10
X-RateLimit-Remaining: 0
X-RateLimit-Reset: 1693123456789
```

Database Operations

Schema Updates

```
# After editing prisma/schema.prisma
npx prisma migrate dev --name "add_new_field"

# Generate client after schema changes
npx prisma generate

# Deploy to production
npx prisma migrate deploy
```

Seed Operations

```
# Seed default rulebook (idempotent)
npm run tsx scripts/seed_rulebook.ts

# Check seeded data
npx prisma studio  # Web UI at http://localhost:5555
```

Quality Framework Integration

Quality Checking Pipeline

```
import { ObservabilityTracker } from '@/lib/observability'
import { processQualityGating } from '@/lib/wordpress'
import { validateI18nCompliance } from '@/lib/i18n'

// Initialize tracking
const tracker = new ObservabilityTracker(
  'pipeline-123',
  'site-abc',
  'Article Title'
)

// Track LLM stage
const llmStage = tracker.startStage('content_generation')
llmStage.complete('gpt-4', 1500, 800, 0.045, true)

// Quality checking
const qualityStage = tracker.startStage('quality_check')
const score = await checkContentQuality(content, rulebook)
qualityStage.complete('quality-checker', 0, 0, 0, true)

// WordPress integration with gating
const wpClient = new WordPressClient(wpConfig)
const result = await processQualityGating(
  wpClient,
  { title, html, score, details: {}, lang: 'en' },
  { ignore_rulebook: false },
  75 // threshold
)

// Finalize observability
await tracker.finalize(score, { wp_post_id: result.postId })
```

i18n/RTL Support

```
import { generateSlug, formatCitation, validateI18nCompliance } from '@/lib/i18n'

// Arabic content handling
const arabicSlug = generateSlug('مثال على المحتوى العربي', 'ar')
// Output: "article-ar-1a2b3c"

const arabicCitation = formatCitation(
  'ويكيبيديا',
  'مقال عن الذكاء الاصطناعي',
  'https://ar.wikipedia.org/wiki/...',
  'ar'
)
// Output: "المصدر: ويكيبيديا. \"مقال عن الذكاء الاصطناعي\". متاح على: ..."

// Validate compliance
const validation = validateI18nCompliance({
  title: 'Arabic Article',
  slug: 'article-ar-1a2b3c',
  html: '<p>Content...</p>',
  lang: 'ar',
  citations: [arabicCitation],
  altTexts: ['صورة توضيحية: مثال'],
  schema: { inLanguage: 'ar' }
})
```

Testing

Manual API Testing

```
# Health check
curl http://localhost:3001/api/health
# Expected: {"ok":true}

# Unauthorized request
curl -X POST http://localhost:3001/api/rulebook
# Expected: 401 Unauthorized

# With valid token
curl -H "Authorization: Bearer $TOKEN" \
     -X POST http://localhost:3001/api/rulebook \
     --data '{"action":"get"}'
# Expected: 200 OK with rulebook data

# Rate limit test
for i in {1..12}; do
  curl -s -o /dev/null -w "%{http_code} " \
    -H "Authorization: Bearer $TOKEN" \
    -X POST http://localhost:3001/api/rulebook
done
# Expected: First 10 return 200, then 429s
```

E2E Quality Pipeline Test

```
# Run quality checker on sample content
cd python
python -m orion.quality.checker \
  --content-file test_content.md \
  --site-id test-site \
  --output results.json

# Check WordPress integration (dev mode)
export NODE_ENV=development
node -e "
  const { processQualityGating } = require('./lib/wordpress');
  // Will output WORDPRESS_STUB logs instead of real API calls
"
```

Database Testing

```
# Test migration status
npx prisma migrate status

# Validate schema
npx prisma validate

# Test database connection
npx prisma db pull --preview-feature
```

Monitoring & Observability

Structured Logging

All operations emit structured JSON logs:

```
{
  "AUDIT_LOG": {
    "route": "/api/rulebook",
    "actor": "admin@example.com",
    "action": "get_rulebook_success",
    "metadata": {"version": 1},
    "timestamp": "2025-08-29T12:00:00.000Z",
    "ip": "127.0.0.1"
  }
}
```

Observability Reports

```
{
  "OBSERVABILITY_REPORT": {
    "pipeline_id": "pipe-123",
    "site_id": "site-abc",
    "total_latency_ms": 5420,
    "total_cost_usd": 0.087,
    "total_tokens": 2300,
    "stages": [{
      "stage": "content_generation",
      "model": "gpt-4",
      "tokens_input": 1500,
      "tokens_output": 800,
      "latency_ms": 3200,
      "cost_usd": 0.045,
      "success": true
    }],
    "quality_score": 82,
    "flags": {"wp_post_id": 456}
  }
}
```

GitHub Actions Integration

Bi-monthly Updates

Schedule: 1st and 15th of each month at 2 AM UTC

Manual Trigger:

```
# Via GitHub UI: Actions → "Bi-monthly Rulebook Update" → Run workflow

# Set options:
# - dry_run: true (test mode)
# - force_update: false (only update if changes detected)
```

Artifacts: Each run produces:
- `rulebook-vN.json` - New rulebook version
- `rulebook-diff.json` - Changes summary
- `rollback-YYYYMMDD.sql` - Rollback script

Rollback Process

1. **Download rollback artifact** from GitHub Actions run
2. **Test rollback** in development:

```
psql $DATABASE_URL -f rollback-20250829.sql
```

1. **Verify API** returns previous rulebook version
2. **Apply to production** if needed

Troubleshooting

Common Issues

Build Errors

```
# Clear build cache
rm -rf .next node_modules/.cache

# Reinstall dependencies
rm -rf node_modules package-lock.json
npm install

# Regenerate Prisma client
npx prisma generate
```

API 404 Errors

- Verify route files exist in `app/api/` (not `pages/api/`)
- Check `export async function GET/POST` syntax
- Restart dev server after route changes

Authentication Failures

```
# Check token in database
npx prisma studio
# Navigate to connections table → console_api_token entry

# Test token encoding
node -e "
const { decryptJson } = require('./lib/crypto');
console.log(decryptJson('ENCRYPTED_TOKEN_FROM_DB'));
"
```

Rate Limiting Issues

```
# Clear rate limit store (dev only)
# Restart server or wait for window to expire

# Check headers for limits:
curl -I -H "Authorization: Bearer $TOKEN" http://localhost:3001/api/rulebook
```

Migration Issues

```
# Check current status
npx prisma migrate status

# Pull latest schema from DB
npx prisma db pull

# Reset development database (CAUTION: loses data)
npx prisma migrate reset --force
```

Log Analysis

Search for specific events:

```
# Authentication failures
grep "auth_failed" logs.txt

# Rate limit hits
grep "rate_limit_exceeded" logs.txt

# Quality scores below threshold
grep "review-needed" logs.txt
```

Parse observability data:

```
grep "OBSERVABILITY_REPORT" logs.txt | jq '.total_cost_usd' | awk
'{sum+=$1} END {print "Total cost: $" sum}'
```

Environment Variables

Required

```
DATABASE_URL=postgresql://...
NEXTAUTH_SECRET=base64-secret-here
ENCRYPTION_KEY=base64-key-here
```

Optional

```
# WordPress integration
WORDPRESS_SITE_URL=https://yoursite.com
WORDPRESS_USERNAME=admin
WORDPRESS_APP_PASSWORD=generated-password

# LLM APIs (for Python integration)
OPENAI_API_KEY=sk-...
PERPLEXITY_API_KEY=pplx-...

# Production optimizations
NODE_ENV=production
DATABASE_MAX_CONNECTIONS=20
```

Production Deployment

Checklist

- [ ] All environment variables configured
- [ ] Database migrations applied (`npx prisma migrate deploy`)
- [ ] Prisma client generated (`npx prisma generate`)
- [ ] Build successful (`npm run build`)
- [ ] Health endpoint responding (`/api/health`)
- [ ] Bearer token authentication working
- [ ] Rate limits configured appropriately
- [ ] Audit logging enabled
- [ ] Default rulebook seeded

Performance Tuning

```
# Database connection pooling
DATABASE_MAX_CONNECTIONS=20
DATABASE_POOL_TIMEOUT=60000

# Rate limiting (for high-traffic)
RATE_LIMIT_REDIS_URL=redis://...  # Switch from memory to Redis

# Observability
OBSERVABILITY_SAMPLING_RATE=0.1   # Sample 10% of requests
```

This completes the Phase 7 developer documentation. All APIs, authentication, database operations, testing procedures, and troubleshooting guides are covered.