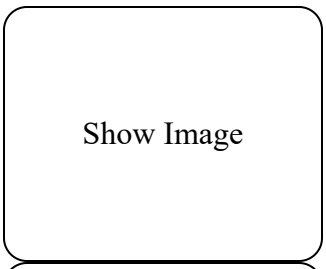
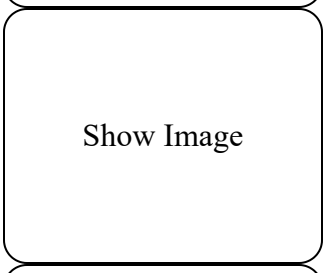
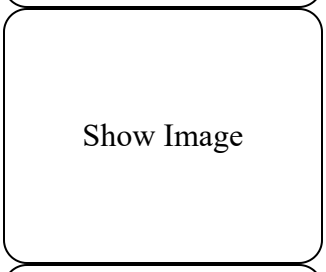
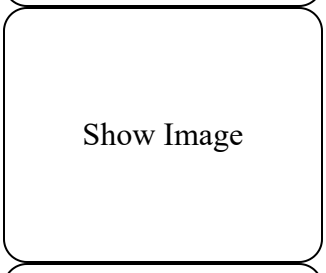
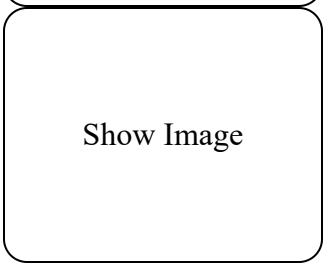
CIE v2.3.2 - Catalog Intelligence Engine











Enterprise-grade product content management system with AI-powered validation, tier-based governance, and automated quality enforcement.

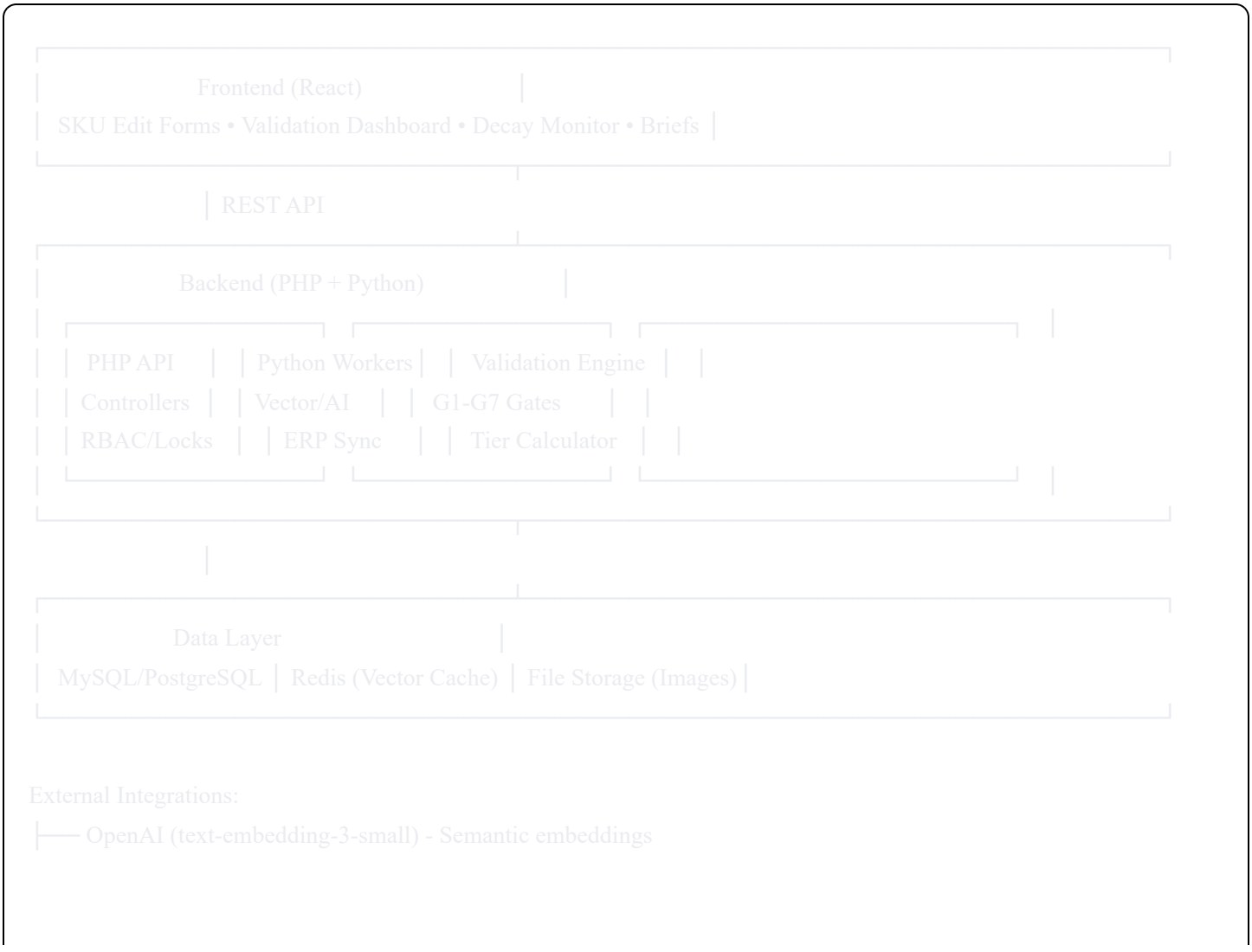
🎯 What is CIE?

The Catalog Intelligence Engine (CIE) is a sophisticated product content management platform that ensures your SKU data meets publication standards through automated validation, semantic clustering, and tier-based content governance.

Core Capabilities

- **7-Gate Validation Pipeline** - Every SKU must pass G1-G7 before publication (basic info, images, SEO, vector similarity, technical specs, commercial data, expert authority)
- **Tier-Based Governance** - Automatic classification into Hero/Support/Harvest/Kill tiers with field-level edit restrictions
- **Vector Semantic Validation** - AI embeddings ensure content matches cluster intent (cosine similarity ≥0.72)
- **Multi-Engine AI Auditing** - Citation checking across Perplexity, OpenAI, Anthropic, and Gemini
- **Automated Decay Detection** - Auto-generates content refresh tasks when Hero SKUs lose AI visibility
- **ERP Integration** - Nightly sync of pricing, margin, and volume data with automatic tier recalculation
- **Role-Based Access Control** - 9 distinct roles with granular permissions (ADMIN, SEO_GOVERNOR, CONTENT_EDITOR, etc.)

🏗️ Architecture Overview



Tech Stack

Backend

- PHP 8.1+ (API layer, business logic)
- Python 3.11+ (Vector validation, AI integrations, ETL jobs)
- MySQL 8.0+ or PostgreSQL 14+ (Primary database)
- Redis 7.0+ (Vector cache, session storage)

Frontend

- React 18.2+
- Zustand (State management)
- React Hook Form (Form validation)
- Recharts (Data visualization)
- Axios (API client)

Infrastructure

- Docker & Docker Compose
- Nginx (Reverse proxy)
- Cron (Scheduled jobs)

🚀 Quick Start

Prerequisites

- Docker & Docker Compose
- PHP 8.1+ with extensions: `pdo`, `pdo_mysql`, `redis`, `gd`, `mbstring`
- Python 3.11+ with `pip`
- Node.js 18+ with `npm`
- MySQL 8.0+ or PostgreSQL 14+
- Redis 7.0+

Installation

1. Clone the repository

```bash
git clone https://github.com/yourcompany/cie-v232.git
cd cie-v232
```

2. Copy environment configuration

```bash
cp .env.example .env
# Edit .env and add your API keys, database credentials
```

3. Start with Docker Compose (Recommended)

```bash
docker-compose up -d
```

This starts:

- PHP-FPM container (API server)
- Python worker container (Vector validation, AI jobs)
- MySQL container
- Redis container
- Nginx container (port 8080)

4. Initialize database

```bash
docker-compose exec php-api bash
cd /app
make migrate    # Run all migrations
make seed       # Seed locked intents, tiers, roles
make test-data  # Load golden test data (optional)
```

5. Install frontend dependencies

```bash
```

```
cd frontend
npm install
npm run dev     # Development server on port 5173
```

6. Access the application

- Frontend: http://localhost:5173
- API: http://localhost:8080/api/v1
- API Docs: http://localhost:8080/api/docs

Default login:

- Username: `admin@company.com`
- Password: `cie_admin_2026` (CHANGE IMMEDIATELY)

📋 Configuration

Required Environment Variables

```bash
```

```
# Application
APP_ENV=production
APP_DEBUG=false
APP_URL=https://cie.yourcompany.com

# Database
DB_CONNECTION=mysql
DB_HOST=db
DB_PORT=3306
DB_DATABASE=cie_v232
DB_USERNAME=cie_user
DB_PASSWORD=your_secure_password

# Redis (Vector Cache)
REDIS_URL=redis://redis:6379/0

# OpenAI (Embeddings - REQUIRED)
OPENAI_API_KEY=sk-proj-...

# AI Audit Engines
PERPLEXITY_API_KEY=pplx-...
ANTHROPIC_API_KEY=sk-ant-...
GEMINI_API_KEY=...

# ERP Integration
ERP_CONNECTION_STRING=Driver={SQL Server};Server=erp.company.com;Database=ProductDB;
ERP_SYNC_ENABLED=true

# Email Notifications
MAIL_HOST=smtp.company.com
MAIL_PORT=587
MAIL_USERNAME=cie@company.com
MAIL_PASSWORD=...

# Vector Validation
SIMILARITY_THRESHOLD=0.72
EMBEDDING_TIMEOUT=3000
VECTOR_CACHE_TTL=86400

# Feature Flags
FAIL_SOFT_ENABLED=true
```

```
DECAY_MONITORING_ENABLED=true
AUTO_BRIEF_GENERATION=true
```

Optional Configuration

Edit `config/*.php` files to customize:

- Database connection pooling
- Cache TTLs
- Rate limiting
- File upload limits
- Email templates

🎓 User Guide

The 9 Locked Intents

These are **hardcoded** and cannot be edited by users:

1. **problem_solving** - Troubleshooting, fixes, solutions
2. **comparison** - Product A vs Product B comparisons
3. **compatibility** - What works with what
4. **product_specs** - Technical specifications
5. **installation** - Setup and installation guides
6. **troubleshooting** - Diagnostic and repair
7. **buyer_guide** - Purchasing decision support
8. **use_case** - Real-world application scenarios
9. **product_overview** - General product information

The 4 Tiers

SKUs are automatically assigned to tiers based on margin and volume:

Tier	Criteria	Edit Rights	Publication Gates
Hero	Top 20% margin+volume OR strategic_hero=true	All fields editable	G1-G7 required
Support	Profitable (margin > threshold)	All fields editable	G1-G6 required, G7 warning
Harvest	Low margin but still selling	Title/images LOCKED	G1-G6 required, G7 skipped
Kill	Negative margin OR no sales 90+ days	ALL fields LOCKED	No publication allowed

The 7 Validation Gates

Every SKU must pass these before publication:

- **G1 - Basic Info**: SKU code, title, short description present
- **G2 - Images**: At least 1 hero image uploaded
- **G3 - SEO**: Meta title (≤60 chars), meta description (≤160 chars)
- **G4 - Vector**: Cosine similarity ≥0.72 to cluster centroid (semantic match)
- **G5 - Technical**: All required specs filled, units consistent
- **G6 - Commercial**: Price and margin data from ERP sync
- **G7 - Expert**: Authority fields completed (Hero/Support only)

G1-G6 are BLOCKING (must pass to publish). **G7 is warning-only** for Harvest/Kill.

Role Permissions

Role	Key Permissions
ADMIN	Everything (tier changes, cluster edits, user management)
SEO_GOVERNOR	Manage clusters, modify intent statements, approve cluster changes
CONTENT_EDITOR	Edit SKU content (subject to tier locks)
CONTENT_LEAD	Edit + assign briefs + view effort reports
PRODUCT_SPECIALIST	Edit expert authority + safety certifications only
CHANNEL_MANAGER	View readiness scores, manage channel mappings
FINANCE	Trigger ERP sync, view tier assignments, recalculate tiers
AI_OPS	Run AI audits, view decay monitor, manage golden queries

Role	Key Permissions
VIEWER	Read-only dashboard access

🔧 Development

Running Tests

Backend PHP Tests

```bash
cd backend/php
composer install
vendor/bin/phpunit tests/
```

Backend Python Tests

```bash
cd backend/python
pip install -r requirements.txt
pytest tests/
```

Frontend Tests

```bash
cd frontend
npm run test          # Unit tests (Vitest)
npm run test:e2e      # E2E tests (Cypress)
```

Integration Tests

```bash
# Run full pipeline test with golden data
./scripts/testing/run_golden_tests.sh
```

Database Migrations

Create new migration

```bash
# Add file: database/migrations/016_your_migration_name.sql
# Run migrations
make migrate
```

Rollback last migration

```bash
make rollback
```

Adding a New Gate

1. Create gate class: `backend/php/src/Validators/Gates/G8_YourGate.php`
2. Implement `GateInterface` with `validate(Sku $sku): GateResult`
3. Add to `GateValidator.php` orchestrator
4. Update enum: `backend/php/src/Enums/GateType.php`
5. Add tests: `backend/php/tests/Unit/G8_YourGateTest.php`

Code Style

PHP

```bash
composer run lint     # Check PSR-12 compliance
composer run format   # Auto-fix style issues
```

Python

```bash
black backend/python/src/     # Format code
pylint backend/python/src/    # Lint checks
```

JavaScript

```bash
npm run lint          # ESLint
npm run format        # Prettier
```

📊 Monitoring & Operations

Health Checks

System Health

```bash
curl http://localhost:8080/api/v1/health
```

Database Connection

```bash
docker-compose exec php-api php artisan db:ping
```

Redis Cache

```bash
docker-compose exec redis redis-cli PING
```

Logs

Application Logs

```bash
docker-compose logs -f php-api      # PHP API logs
docker-compose logs -f python-worker # Python job logs
```

Validation Logs (G1-G7 results)

```sql
SELECT * FROM validation_logs
WHERE sku_id = 'your-sku-uuid'
ORDER BY created_at DESC
LIMIT 10;
```

Tier Change Audit

```sql
```

```
SELECT * FROM tier_history
WHERE sku_id = 'your-sku-uuid'
ORDER BY changed_at DESC;
```

Scheduled Jobs

View cron schedule

```bash
crontab -l
```

Manually trigger jobs

```bash
# ERP Sync (normally runs at 2 AM daily)
./jobs/nightly_erp_sync.sh

# Decay Check (normally runs weekly)
./jobs/weekly_decay_check.sh

# Vector Retry Queue (normally runs hourly)
./jobs/hourly_vector_retry.sh
```

Performance Metrics

Key Metrics to Monitor

Metric	Target	Alert Threshold
Vector Cache Hit Rate	>80%	<70%
Validation API p95	<500ms	>1000ms
ERP Sync Duration	<10min	>20min
Gate G4 Timeout Rate	<5%	>10%
Daily Validation Volume	-	Track trends

Grafana Dashboards

- System Health: http://localhost:3000/d/cie-health

- Gate Validation Metrics: http://localhost:3000/d/cie-gates
- Tier Distribution: http://localhost:3000/d/cie-tiers

Security

Authentication

CIE uses **JWT tokens** for API authentication:

1. Login: `POST /api/v1/auth/login` returns JWT token
2. Include in requests: `Authorization: Bearer <token>`
3. Token expiry: 8 hours (configurable)
4. Refresh: `POST /api/v1/auth/refresh`

RBAC Implementation

All endpoints check permissions via `RBACMiddleware`:

```php
// Example: Only FINANCE can trigger tier recalculation
Route::post('/tiers/recalculate', [TierController::class, 'recalculate'])
    ->middleware(['auth', 'rbac:FINANCE,ADMIN']);
```

Data Protection

- **API Keys**: Stored in `.env`, never committed to git
- **Database Passwords**: Rotated quarterly
- **File Uploads**: Validated for type, size, scanned for malware
- **SQL Injection**: All queries use parameterized statements
- **XSS Protection**: All user input sanitized before rendering

Audit Trail

Every change is logged in `audit_log` table:

- Who made the change (user_id)
- What changed (entity_type, entity_id, field_name)
- Old and new values
- Timestamp

- IP address

```sql
SELECT * FROM audit_log
WHERE entity_type = 'sku'
  AND entity_id = 'your-sku-uuid'
ORDER BY created_at DESC;
```

🚢 Deployment

Production Deployment Checklist

- [ ] Environment variables set (all API keys, DB credentials)
- [ ] Database migrations run on production DB
- [ ] Redis configured with persistence enabled
- [ ] ERP connection tested and sync scheduled
- [ ] SSL certificates installed (HTTPS required)
- [ ] Firewall rules configured (allow ports 80, 443 only)
- [ ] Backup cron job scheduled (daily DB backups)
- [ ] Monitoring alerts configured (PagerDuty/Opsgenie)
- [ ] Log aggregation enabled (ELK stack or equivalent)
- [ ] Load testing completed (500 req/s target)

Deployment Scripts

Staging Deployment

```bash
./scripts/deployment/deploy_staging.sh
```

Production Deployment

```bash
./scripts/deployment/deploy_production.sh
```

Rollback

```bash
```

```
./scripts/deployment/rollback.sh
```

Database Backup & Restore

Backup

```bash
./scripts/maintenance/backup_database.sh
# Creates timestamped backup in /backups/
```

Restore

```bash
./scripts/maintenance/restore_database.sh /backups/cie_backup_20260213.sql
```

📖 API Documentation

Interactive API docs available at: http://localhost:8080/api/docs

Key Endpoints

SKU Management

- `GET /api/v1/skus` - List all SKUs (paginated)
- `GET /api/v1/skus/{id}` - Get SKU details
- `POST /api/v1/skus` - Create new SKU
- `PUT /api/v1/skus/{id}` - Update SKU (tier-lock enforced)
- `DELETE /api/v1/skus/{id}` - Delete SKU (soft delete)

Validation

- `POST /api/v1/skus/{id}/validate` - Run all 7 gates
- `GET /api/v1/validation/history/{sku_id}` - Get validation history

Tier Management

- `POST /api/v1/tiers/recalculate` - Recalculate all tiers (FINANCE only)
- `GET /api/v1/tiers/distribution` - View tier breakdown

AI Audit

- `POST /api/v1/audit/{sku_id}` - Run AI citation audit
- `GET /api/v1/audit/decay-risks` - List SKUs at risk of decay

Content Briefs

- `GET /api/v1/briefs` - List all briefs
- `POST /api/v1/briefs` - Create manual brief
- `PUT /api/v1/briefs/{id}` - Update brief status

Clusters

- `GET /api/v1/clusters` - List all clusters
- `POST /api/v1/clusters` - Create cluster (SEO_GOVERNOR only)
- `PUT /api/v1/clusters/{id}` - Update cluster intent

Example API Call

bash

```
# Validate a SKU
curl -X POST http://localhost:8080/api/v1/skus/550e8400-e29b-41d4-a716-446655440000/validate \
  -H "Authorization: Bearer your-jwt-token" \
  -H "Content-Type: application/json"

# Response
{
  "sku_id": "550e8400-e29b-41d4-a716-446655440000",
  "overall_status": "INVALID",
  "can_publish": false,
  "gates": [
    {"gate": "G1_BASIC_INFO", "passed": true, "reason": "All required fields present"},
    {"gate": "G2_IMAGES", "passed": true, "reason": "1 hero image uploaded"},
    {"gate": "G3_SEO", "passed": true, "reason": "Meta fields within limits"},
    {"gate": "G4_VECTOR", "passed": false, "reason": "Similarity 0.65 < threshold 0.72", "blocking": true},
    {"gate": "G5_TECHNICAL", "passed": true, "reason": "All specs valid"},
    {"gate": "G6_COMMERCIAL", "passed": true, "reason": "Price and margin synced from ERP"},
    {"gate": "G7_EXPERT", "passed": false, "reason": "Authority fields incomplete", "blocking": false}
  ],
  "next_action": "Revise description to match cluster intent or request cluster reassignment"
}
```

🤝 Contributing

Development Workflow

1. Create feature branch: `git checkout -b feature/your-feature-name`
2. Make changes
3. Write tests (unit + integration)
4. Run linters: `make lint`
5. Run tests: `make test`
6. Commit: `git commit -m "feat: your feature description"`
7. Push: `git push origin feature/your-feature-name`
8. Open Pull Request

Commit Message Format

Follow Conventional Commits:

```
feat: add G8 validation gate for sustainability certifications
fix: resolve tier lock bypass bug in edit form
docs: update API documentation for vector validation
refactor: optimize vector cache hit rate
test: add integration tests for ERP sync
```

PR Review Checklist

- [ ] All tests pass (`make test`)
- [ ] Code coverage >80%
- [ ] Linting passes (`make lint`)
- [ ] Documentation updated (if API changed)
- [ ] Migration file added (if schema changed)
- [ ] Security review (if touching auth/RBAC)
- [ ] Performance impact assessed (if touching hot paths)

🐛 Troubleshooting

Common Issues

Issue: Vector validation always returns DEGRADED

- **Cause**: OpenAI API key missing or invalid
- **Fix**: Check `OPENAI_API_KEY` in `.env`, verify key is valid at https://platform.openai.com/api-keys

Issue: ERP sync fails with "Connection timeout"

- **Cause**: Firewall blocking connection to ERP server
- **Fix**: Check `ERP_CONNECTION_STRING`, ensure ports 1433 (SQL Server) or 3306 (MySQL) are open

Issue: Tier locks not enforced in edit form

- **Cause**: Frontend not calling RBAC check before enabling fields
- **Fix**: Check `useTierLock()` hook is applied to form component

Issue: Images not uploading

- **Cause**: Upload directory not writable
- **Fix**: `chmod 755 storage/uploads/images/`

Issue: Redis cache always misses

- **Cause**: Redis connection failed silently
- **Fix**: Check `REDIS_URL`, verify Redis is running: `docker-compose ps redis`

Debug Mode

Enable debug mode in `.env`:

```bash
APP_DEBUG=true
LOG_LEVEL=debug
```

Warning: Never enable in production (exposes sensitive data in error messages)

Getting Help

1. Check logs: `docker-compose logs -f`
2. Search existing issues: https://github.com/yourcompany/cie-v232/issues
3. Create new issue with:
    - Steps to reproduce
    - Expected vs actual behavior
    - Logs and error messages

- Environment details (PHP version, OS, etc.)

📜 License

Proprietary License - All rights reserved.

This software is the property of [Your Company Name]. Unauthorized copying, distribution, or modification is strictly prohibited.

For licensing inquiries: licensing@yourcompany.com

🙏 Acknowledgments

Built with:

- OpenAI - Semantic embeddings
- Anthropic Claude - AI audit engine
- Perplexity - Citation checking
- Google Gemini - Multi-engine validation

📞 Support

- **Documentation**: https://docs.cie.yourcompany.com
- **Email**: cie-support@yourcompany.com
- **Slack**: #cie-support (internal)
- **Runbook**: docs/deployment/runbook.md

Version 2.3.2 | February 2026 | Built with ❤️ by the CIE Team