Smart Search - User Guide

> Intelligent Source Code Navigation System V2.1

Mục lục

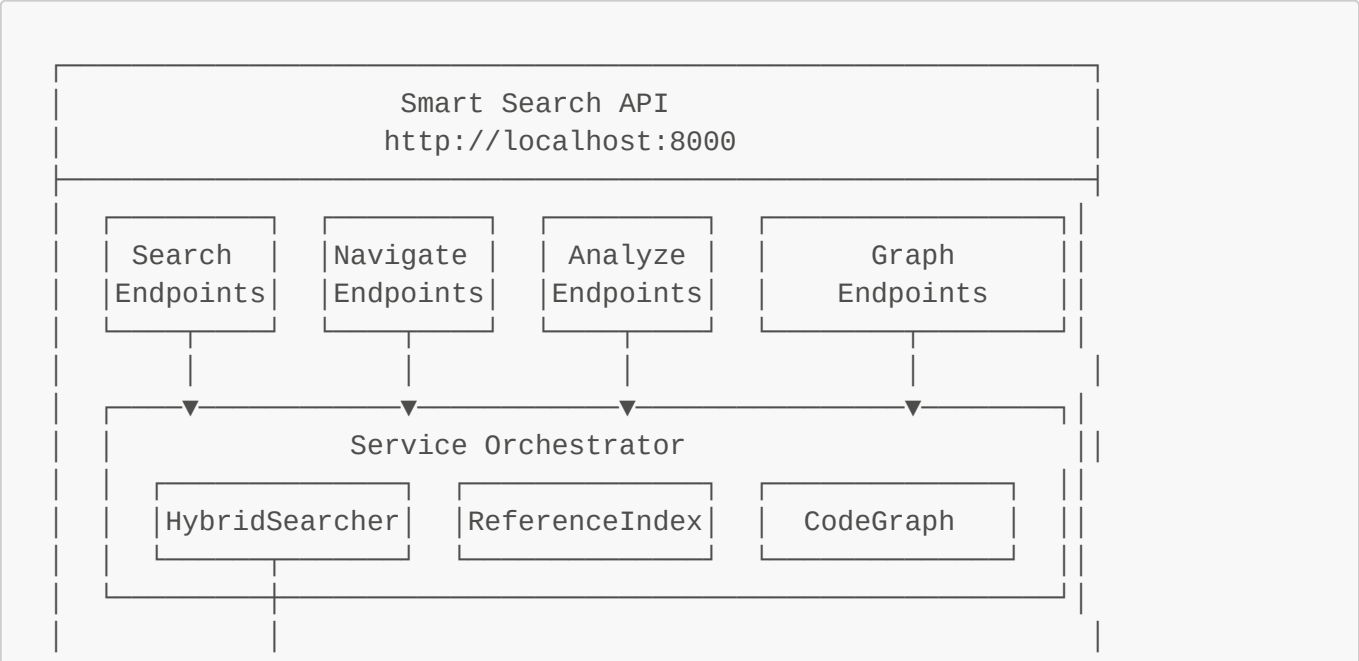
---

1. Giới thiệu

Smart Search là hệ thống tìm kiếm và điều hướng mã nguồn thông minh, kết hợp:

- **Keyword Search**: Tìm kiếm văn bản truyền thống
- **Semantic Search**: Tìm kiếm dựa trên ngữ nghĩa bằng AI
- **Hybrid Search**: Kết hợp cả hai phương pháp
- **Code Graph**: Phân tích quan hệ giữa các thành phần code
- **RAG (Retrieval-Augmented Generation)**: Trả lời câu hỏi bằng ngôn ngữ tự nhiên

Kiến trúc hệ thống



```
│   ┌──────────────▼──────────────┐                                   │
│   │    Meilisearch             │  (Vector Search + Full-text)      │
│   │   localhost:7700           │                                   │
│   └────────────────────────────┘                                   │
└────────────────────────────────────────────────────────────────────┘
```

---

2. Quick Start

Yêu cầu

- Python 3.12+
- Docker & Docker Compose
- 4GB+ RAM

Bước 1: Khởi động Meilisearch

```bash
cd /home/fong/Projects/smart_search

# Start Meilisearch
docker compose up -d meilisearch

# Bật vector store (chạy 1 lần sau khi khởi động)
curl -X PATCH http://localhost:7700/experimental-features \
  -H "Authorization: Bearer smart_search_dev_key" \
  -H "Content-Type: application/json" \
  -d '{"vectorStore": true}'
```

Bước 2: Khởi động Smart Search API

```bash
# Bật đầy đủ tính năng
FF_USE_HYBRID_SEARCHER=true \
FF_USE_REFERENCE_INDEX=true \
MEILISEARCH_API_KEY=smart_search_dev_key \
PYTHONPATH=src .venv/bin/uvicorn smart_search.api:create_app \
  --factory --host 0.0.0.0 --port 8000
```

Bước 3: Kiểm tra

```bash
# Kiểm tra API
curl http://localhost:8000/

# Kiểm tra health
curl http://localhost:8000/health
```

Kết quả mong đợi:

```
{
  "status": "healthy",
  "services": {
    "graph": true,
    "searcher": true,
    "indexer": true,
    "meilisearch": {"available": true, "healthy": true}
  }
}
```

---

3. Tính năng

3.1 Search - Tìm kiếm mã nguồn

Keyword Search (Tìm kiếm từ khóa)

```
curl -X POST http://localhost:8000/api/v1/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "authentication",
    "limit": 10,
    "search_type": "keyword"
  }'
```

Semantic Search (Tìm kiếm ngữ nghĩa)

Tìm kiếm dựa trên ý nghĩa, không chỉ từ khóa:

```
curl -X POST http://localhost:8000/api/v1/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "function that validates user input",
    "limit": 10,
    "search_type": "semantic"
  }'
```

Hybrid Search (Khuyên dùng)

Kết hợp keyword + semantic:

```
curl -X POST http://localhost:8000/api/v1/search \
  -H "Content-Type: application/json" \
```

```
  -d '{
    "query": "parse JSON config",
    "limit": 10
  }'
```

Search với Filters

```
curl -X POST http://localhost:8000/api/v1/search \
  -H "Content-Type: application/json" \
  -d '{
    "query": "parse",
    "limit": 20,
    "language": "python",
    "code_type": "function",
    "file_path": "src/"
  }'
```

GET Search (Đơn giản)

```
curl "http://localhost:8000/api/v1/search?q=authenticate&limit=5"
```

Tham số:

Tham số	Kiểu	Mặc định	Mô tả
`query`	string	bắt buộc	Từ khóa tìm kiếm (1-1000 ký tự)
`search_type`	string	`hybrid`	`keyword`, `semantic`, `hybrid`
`limit`	int	20	Số kết quả tối đa (1-100)
`offset`	int	0	Offset cho phân trang
`language`	string	null	Lọc theo ngôn ngữ (python, php, ...)
`file_path`	string	null	Lọc theo đường dẫn file
`code_type`	string	null	Lọc theo loại (function, class, method)

Response:

```
{
  "query": "parse",
  "total_hits": 15,
  "hits": [
    {
      "id": "src.parser.parse_json",
      "name": "parse_json",
```

```
      "qualified_name": "src.parser.parse_json",
      "code_type": "function",
      "file_path": "src/parser.py",
      "line_start": 45,
      "line_end": 67,
      "content": "def parse_json(data: str) -> dict:\n    ...",
      "language": "python",
      "score": 0.95,
      "highlights": ["<mark>parse</mark>_json"]
    }
  ],
  "processing_time_ms": 12.5
}
```

3.2 RAG Search - Tìm kiếm AI

RAG Query (Trả lời ngôn ngữ tự nhiên)

```
curl -X POST http://localhost:8000/api/v1/search/rag \
  -H "Content-Type: application/json" \
  -d '{
    "query": "How does the authentication flow work?",
    "mode": "hybrid",
    "include_explanation": true
  }'
```

Modes:

Mode	Mô tả	Phù hợp cho
`local`	Tìm trong context cục bộ	Câu hỏi về function/class cụ thể
`global`	Tìm toàn bộ codebase	Câu hỏi về kiến trúc
`hybrid`	Kết hợp local + global	Câu hỏi chung (khuyên dùng)
`drift`	Theo dõi quan hệ code	"X kết nối với Y như thế nào?"

Response:

```
{
  "query": "How does authentication work?",
  "mode": "hybrid",
  "answer": "Authentication được xử lý bởi function `authenticate()` trong
`src/auth.py`. Nó validate credentials với database và trả về JWT
token...",
  "contexts": [
    {
      "code_id": "src.auth.authenticate",
```

```
      "content": "def authenticate(username, password)...",
      "relevance_score": 0.92
    }
  ],
  "citations": ["auth.authenticate", "auth.verify_token"],
  "processing_time_ms": 1250.5
}
```

Find Similar Code (Tìm code tương tự)

```
curl -X POST http://localhost:8000/api/v1/search/similar \
  -H "Content-Type: application/json" \
  -d '{
    "code": "def validate_email(email):\n    import re\n    pattern =
r\"^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\\.[a-zA-Z0-9-.]+$\"\n    return
re.match(pattern, email) is not None",
    "limit": 5
  }'
```

Ứng dụng:

- Tìm code trùng lặp
- Khám phá implementations tương tự
- Cơ hội refactor

Search Suggestions (Autocomplete)

```
curl "http://localhost:8000/api/v1/search/suggest?q=auth&limit=10"
```

3.3 Navigate - Điều hướng code

Go to Definition

```
curl -X POST http://localhost:8000/api/v1/navigate/definition \
  -H "Content-Type: application/json" \
  -d '{
    "symbol": "authenticate",
    "file_path": "src/api/login.py",
    "line": 25
  }'
```

Find References (Tìm nơi sử dụng)

```
curl "http://localhost:8000/api/v1/navigate/references/{code_id}?limit=50"
```

Response:

```
{
  "code_id": "auth.authenticate",
  "references": [
    {"file_path": "src/api/login.py", "line": 25, "context": "user =
authenticate(username, password)"},
    {"file_path": "src/api/oauth.py", "line": 42, "context": "return
authenticate(email, token)"}
  ],
  "total": 2
}
```

Find Callers (Ai gọi function này?)

```
curl "http://localhost:8000/api/v1/navigate/callers/{code_id}"
```

Find Callees (Function này gọi gì?)

```
curl "http://localhost:8000/api/v1/navigate/callees/{code_id}"
```

Get Class Hierarchy

```
curl "http://localhost:8000/api/v1/navigate/hierarchy/{code_id}?depth=3"
```

File Outline (Danh sách symbols trong file)

```
curl "http://localhost:8000/api/v1/navigate/outline?file_path=src/auth.py"
```

Response:

```
{
  "file_path": "src/auth.py",
  "symbols": [
    {"name": "AUTH_SECRET", "type": "constant", "line": 5},
    {"name": "authenticate", "type": "function", "line": 15},
```

```
    {"name": "AuthError", "type": "class", "line": 72}
  ]
}
```

Search Symbols

```
curl "http://localhost:8000/api/v1/navigate/symbols?query=User&type=class"
```

3.4 Analyze - Phân tích code

Explain Code (Giải thích code)

```
curl -X POST http://localhost:8000/api/v1/analyze/explain \
  -H "Content-Type: application/json" \
  -d '{
    "code": "def factorial(n):\n    return 1 if n <= 1 else n *
factorial(n-1)",
    "question": "What is the time complexity?"
  }'
```

Impact Analysis (Phân tích ảnh hưởng)

```
curl -X POST http://localhost:8000/api/v1/analyze/impact \
  -H "Content-Type: application/json" \
  -d '{
    "code_id": "auth.authenticate",
    "depth": 3
  }'
```

Response:

```
{
  "code_id": "auth.authenticate",
  "impact": {
    "direct_dependents": 5,
    "indirect_dependents": 23,
    "affected_files": ["api/login.py", "api/oauth.py"],
    "risk_level": "high"
  }
}
```

Code Metrics (Độ phức tạp)

```
curl "http://localhost:8000/api/v1/analyze/metrics/{code_id}"
```

Get Dependencies

```
curl "http://localhost:8000/api/v1/analyze/dependencies/{code_id}?
direction=both&depth=2"
```

Find Duplicates (Tìm code trùng lặp)

```
curl "http://localhost:8000/api/v1/analyze/duplicates/{code_id}?
threshold=0.8"
```

Summarize File

```
curl -X POST http://localhost:8000/api/v1/analyze/summarize \
  -H "Content-Type: application/json" \
  -d '{"file_path": "src/auth.py"}'
```

3.5 Graph - Code Graph Operations

Graph Statistics

```
curl http://localhost:8000/api/v1/graph/stats
```

Get/List Nodes

```
# Get single node
curl "http://localhost:8000/api/v1/graph/node/{node_id}"

# List nodes
curl "http://localhost:8000/api/v1/graph/nodes?type=function&limit=50"
```

Get/List Edges

```
curl "http://localhost:8000/api/v1/graph/edges?type=calls&limit=50"
```

Get Subgraph

```
curl "http://localhost:8000/api/v1/graph/subgraph/{node_id}?
depth=2&max_nodes=100"
```

Find Path (Tìm đường đi)

```
curl "http://localhost:8000/api/v1/graph/path?from={node_1}&to=
{node_2}&max_depth=5"
```

Get Communities

```
curl http://localhost:8000/api/v1/graph/communities
```

List/Get Files

```
# List files
curl "http://localhost:8000/api/v1/graph/files?pattern=src/**/*.py"

# Get file nodes
curl "http://localhost:8000/api/v1/graph/file/src%2Fauth.py/nodes"
```

3.6 Index - Quản lý Index

Index Project

```
curl -X POST http://localhost:8000/api/v1/index \
  -H "Content-Type: application/json" \
  -d '{
    "paths": ["/path/to/project"],
    "excludes": ["node_modules", "__pycache__", ".git"],
    "languages": ["python", "php"],
    "incremental": true
  }'
```

Index Status

```
curl http://localhost:8000/api/v1/index/status
```

Get Indexed Files

```
curl "http://localhost:8000/api/v1/index/files?limit=100"
```

Get References (Fast O(1))

```
curl "http://localhost:8000/api/v1/index/references?
filename=auth.py&limit=50"
```

---

4. Cấu hình

Environment Variables

Biến	Mặc định	Mô tả
`MEILISEARCH_URL`	`http://localhost:7700`	URL Meilisearch
`MEILISEARCH_API_KEY`	`smart_search_dev_key`	API key
`DATA_DIR`	`.smart_search`	Thư mục lưu data

Feature Flags

Flag	Mặc định	Mô tả
`FF_USE_HYBRID_SEARCHER`	`false`	Bật HybridSearcher
`FF_USE_REFERENCE_INDEX`	`false`	Bật Reference Index (O(1))
`FF_USE_FILE_CACHE`	`true`	Bật LRU file cache
`FF_USE_ASYNC_IO`	`true`	Bật async file I/O

Ví dụ Production

```
# .env file
MEILISEARCH_URL=http://meilisearch:7700
MEILISEARCH_API_KEY=your_secure_key
DATA_DIR=/var/lib/smart_search
```

```
FF_USE_HYBRID_SEARCHER=true
FF_USE_REFERENCE_INDEX=true
FF_USE_FILE_CACHE=true
FF_USE_ASYNC_IO=true
```

---

5. Load Testing

Interactive Mode

```
locust -f scripts/load_test.py --host=http://localhost:8000
# Mở http://localhost:8089
```

Headless Mode (CI/CD)

```
locust -f scripts/load_test.py \
  --host=http://localhost:8000 \
  --users 100 \
  --spawn-rate 10 \
  --run-time 5m \
  --headless
```

Performance Targets

Metric	Target
Search p50	< 20ms
Search p99	< 50ms
find_references p50	< 100ms
find_references p99	< 500ms
Concurrent users	100+

---

6. Troubleshooting

Meilisearch lỗi kết nối

```
# Kiểm tra Meilisearch
curl http://localhost:7700/health

# Restart
docker compose restart meilisearch
```

Vector Store chưa bật

```
curl -X PATCH http://localhost:7700/experimental-features \
  -H "Authorization: Bearer smart_search_dev_key" \
  -H "Content-Type: application/json" \
  -d '{"vectorStore": true}'
```

Search trả về empty

1. Kiểm tra index:

   ```
   curl http://localhost:8000/api/v1/index/status
   ```

2. Re-index:

   ```
   curl -X POST http://localhost:8000/api/v1/index \
     -H "Content-Type: application/json" \
     -d '{"paths": ["/your/project"], "incremental": false}'
   ```

API Documentation

- **Swagger UI**: http://localhost:8000/docs
- **ReDoc**: http://localhost:8000/redoc
- **OpenAPI JSON**: http://localhost:8000/openapi.json

Smart Search V2.1 - Built with FastAPI, Meilisearch, and AI