

🎯 FlowNote - AI 기반 PARA 분류 시스템

한국어 | English

당신의 문서를 AI가 자동으로 분류합니다

PARA 방식 (Projects, Areas, Resources, Archives)으로 스마트하게 정리



📖 목차

1. [프로젝트 소개](#)
2. [핵심 기능](#)
3. [기술 스택](#)
4. [프로젝트 구조](#)
5. [테스팅](#)
6. [설치 및 실행](#)
7. [사용 방법](#)
8. [개발 히스토리](#)
9. [로드맵](#)
10. [FAQ](#)
11. [기여하기](#)
12. [라이선스](#)
13. [개발자](#)

1. 📖 프로젝트 소개

FlowNote는 AI 기반 문서 자동 분류 시스템입니다. 사용자의 직업과 관심 영역을 학습하여, 업로드된 문서를 PARA 방식으로 지능적으로 분류합니다.

💡 핵심 아이디어

사용자 온보딩 (직업, 관심 영역)

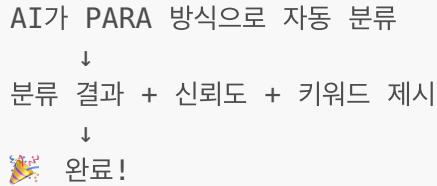


GPT-4o가 사용자 맥락 학습



파일 업로드 (PDF, TXT, MD)





예시:

- **입력:** "프로젝트 제안서_2025.pdf"
- **결과:** 📁 Projects (신뢰도 95%)
- **키워드:** 프로젝트, 마감일, 목표

2. ✨ 핵심 기능

2.1 🚀 스마트 온보딩

- GPT-4o가 사용자 직업 분석
- 10개 영역 추천 → 5개 선택
- 사용자 맥락 저장 및 활용

2.2 📁 AI 기반 자동 분류

- ▣ **Projects** (프로젝트)
→ 기한/마감일이 있는 구체적 목표
예: "11월까지 대시보드 구현"
- ⌚ **Areas** (분야)
→ 지속적 책임 영역
예: "팀 성과 관리 지속 업무"
- 📚 **Resources** (자료)
→ 참고용 정보/학습 자료
예: "Python 최적화 가이드"
- 📦 **Archives** (보관)
→ 완료된 프로젝트 보관
예: "2024년 프로젝트 결과"

2.3 🔎 키워드 검색 시스템

- FAISS 벡터 검색 엔진
- OpenAI Embeddings 활용
- 실시간 유사도 점수 표시
- 검색 히스토리 저장

2.4 📊 실시간 대시보드

- PARA 분류 통계 시각화

- 파일 트리 구조 표시
- 최근 활동 로그
- 메타데이터 관리

2.5 🌐 맥락 기반 분류

- 사용자 직업/관심 영역 반영
- 신뢰도 점수 (0-100%)
- 키워드 태그 자동 생성
- 분류 근거 설명

2.6 🤖 지능형 자동화 시스템

- 자동 재분류: 매일/매주 분류 신뢰도가 낮은 문서를 AI가 재검토
- 스마트 아카이빙: 장기간(90일 이상) 수정되지 않은 Projects 문서를 Archives로 이동 제안
- 정기 리포트: 주간/월간 분류 통계 및 인사이트 리포트 생성
- 시스템 모니터링: 백그라운드 작업 상태 및 동기화 무결성 자동 점검
- Celery & Redis: 안정적인 분산 작업 큐 처리

2.7 🔗 MCP 서버 & Obsidian 연동 (v5.0 Phase 1)

- **Model Context Protocol (MCP)**: Claude Desktop과 통합 가능한 MCP 서버 구현
- **Obsidian 동기화**: 실시간 Vault 파일 감지 및 자동 분류
- **충돌 해결**: 3-way 충돌 감지 및 자동 해결 (Rename 전략)
- **MCP Tools**: `classify_content`, `search_notes`, `get_automation_stats`
- **MCP Resources**: PARA 카테고리별 파일 리스트 제공

2.8 📊 Next.js 기반 모던 대시보드 (v5.0 Phase 2-3)

- **Sync Monitor**: Obsidian 연결 상태 및 MCP 서버 상태 실시간 모니터링
- **PARA Graph View**: React Flow 기반 파일-카테고리 관계 시각화
 - 노드 클릭 인터랙션 (Toast 알림)
 - Deterministic Layout (새로고침 시에도 위치 유지)
 - Zoom/Pan 지원
- **Advanced Stats**: Recharts 기반 통계 차트
 - Activity Heatmap (GitHub 스타일 연간 활동)
 - Weekly Trend (12주 파일 처리량)
 - PARA Distribution (카테고리별 비중 Pie Chart)
- **Mobile Responsive**: 데스크탑/모바일 자동 전환 내비게이션
- **Accessibility**: ARIA 속성 및 스크린 리더 지원

2.9 ⚡ WebSocket 실시간 업데이트 (v6.0 Phase 1)

- **실시간 동기화**: Polling 방식 제거, WebSocket 기반 양방향 통신
- **이벤트 기반 업데이트**: 파일 분류, 동기화 상태 변경 시 즉시 UI 반영
- **네트워크 최적화**: 트래픽 50% 이상 감소
- **연결 관리**: 자동 재연결, Heartbeat 메커니즘
- **타입 안전성**: TypeScript 기반 이벤트 타입 정의

2.10 🔎 Conflict Diff Viewer (v6.0 Phase 2)

- **시각적 비교:** Monaco Editor 기반 Side-by-Side Diff 뷰어
- **3가지 해결 옵션:** Keep Local / Keep Remote / Keep Both
- **Markdown 프리뷰:** 총돌 파일의 렌더링된 결과 미리보기
- **Syntax Highlighting:** 파일 타입별 구문 강조
- **인라인 Diff:** 변경사항 라인별 하이라이트

2.11 🌎 다국어 지원 (i18n) (v6.0 Phase 3)

- **한국어/영어 완벽 지원:** next-intl 기반 다국어 시스템
- **동적 언어 전환:** URL 기반 라우팅 (</ko/dashboard>, </en/dashboard>)
- **SEO 최적화:** 언어별 메타데이터 및 sitemap
- **Backend API i18n:** Accept-Language 헤더 기반 응답 현지화
- **날짜/숫자 포맷:** 로케일별 자동 포맷팅

3. █ 기술 스택

3.1 Backend

기술	버전	용도
Python	3.11.10	개발 언어
FastAPI	0.120.4	REST API 프레임워크
WebSocket	-	실시간 양방향 통신 (v6.0)
LangChain	1.0.2	AI 체인 관리
Celery	5.4.0	비동기 작업 큐 & 스케줄링
Redis	7.x	메시지 브로커 & 캐시
Flower	2.0.1	Celery 모니터링 대시보드
SQLite	3	메타데이터 저장
Uvicorn	0.38.0	ASGI 서버

3.2 Frontend

기술	버전	용도
Next.js	16.1.1	React 프레임워크 (App Router)
React	19.2.3	UI 라이브러리
TypeScript	5.x	타입 안전성
Tailwind CSS	^4.x	스타일링
Shadcn UI	latest	UI 컴포넌트 라이브러리

기술	버전	용도
React Flow	^11.11.4	그래프 시각화
Recharts	^3.6.0	차트 라이브러리
Sonner	^2.0.7	Toast 알림
next-intl	^3.x	다국어 지원 (v6.0)
Monaco Editor	^0.52.x	Diff Viewer (v6.0)

3.3 LLM & AI

기술	모델	용도
OpenAI API	GPT-4o	PARA 분류, 영역 추천
OpenAI API	GPT-4o-mini	경량 작업
OpenAI Embeddings	text-embedding-3-small	벡터 임베딩

3.4 검색 & 데이터

기술	버전	용도
FAISS	1.12.0	벡터 검색 엔진
pdfplumber	0.11.0	PDF 파싱
python-dotenv	1.1.1	환경변수 관리

4. 프로젝트 구조

```
flownote-mvp/
    ├── requirements.txt                                # Python 의존성
    ├── .env.example                                    # 환경변수 예시
    └── .gitignore

    ├── backend/
    │   ├── main.py                                     # FastAPI 백엔드
    │   ├── celery_app/
    │   │   ├── celery.py                               # Celery 설정
    │   │   ├── config.py                             # Celery 인스턴스
    │   │   └── tasks/                                 # 비동기 작업들 (재분류, 아카이빙 등)
    │
    │   └── mcp/
    │       ├── server.py                            # MCP 서버 ✨
    │       └── tools/                                # MCP Tools (classify, search 등)
    │
    └── api/
        └── endpoints/                                # API 엔드포인트
```

└── websocket.py	# WebSocket 엔드포인트 (v6.0) ✨
└── sync.py	# 동기화 & Diff API (v6.0) ✨
└── ...	
└── deps.py	# 의존성 (i18n 로케일 추출) ✨
└── exceptions.py	# 다국어 예외 처리 (v6.0) ✨
└── services/	# 비즈니스 로직 (Service Layer)
└── obsidian_sync.py	# Obsidian 동기화 ✨
└── conflict_resolution_service.py	# 충돌 해결 ✨
└── websocket_manager.py	# WebSocket 연결 관리 (v6.0) ✨
└── diff_service.py	# Diff 생성 (v6.0) ✨
└── i18n_service.py	# 다국어 메시지 (v6.0) ✨
└── ...	
└── core/	# 핵심 설정 (v6.0) ✨
└── config.py	# 애플리케이션 설정
└── embedding.py	# 임베딩 생성
└── faiss_search.py	# FAISS 검색
└── classifier/	# PARA 분류 로직
└── ...	
└── web_ui/	# Next.js Frontend ✨
└── src/	
└── app/	
└── [locale]/	# 다국어 라우팅 (v6.0) ✨
└── page.tsx	# Dashboard
└── graph/page.tsx	# Graph View
└── stats/page.tsx	# Statistics
└── not-found.tsx	# 404 페이지 (i18n) ✨
└── components/	# React 컴포넌트
└── dashboard/	
└── SyncMonitor.tsx	# WebSocket 기반 (v6.0) ✨
└── para/GraphView.tsx	# Graph View
└── conflict/	# Conflict Diff Viewer (v6.0) ✨
└── DiffViewer.tsx	
└── ConflictResolver.tsx	
└── layout/	
└── LanguageSwitcher.tsx	# 언어 전환 (v6.0) ✨
└── i18n/	# 다국어 설정 (v6.0) ✨
└── config.ts	
└── locales/	# 번역 파일 (v6.0) ✨
└── ko.json	
└── en.json	
└── hooks/	# Custom Hooks
└── useWebSocket.ts	# WebSocket Hook (v6.0) ✨
└── config/	# 설정 파일
└── middleware.ts	# next-intl 미들웨어 (v6.0) ✨
└── package.json	
└── data/	# 데이터 저장소
└── docs/	# 문서
└── P/	# 프로젝트 페이지 문서
└── v5_phase1_mcp_server/	# MCP 서버 문서

```

    └── v5_phase2_frontend/          # Frontend 문서
    └── v5_phase3_visualization/    # Visualization 문서
    └── v6.0_phase1_websocket/      # WebSocket 문서 (v6.0) ✨
    └── v6.0_phase2_diff_viewer/    # Diff Viewer 문서 (v6.0) ✨
    └── v6.0_phase3_i18n/          # i18n 문서 (v6.0) ✨
    └── README.md                  # 본 문서 (한국어)
    └── README_EN.md               # 영문 문서

```

5. 🖊 테스트 및 품질 관리

FlowNote는 엄격한 테스트와 품질 관리를 통해 안정성을 보장합니다.

5.1 테스트 실행

```

# 전체 테스트 실행
pytest

# 커버리지 리포트 생성
pytest --cov=backend --cov-report=term-missing

```

5.2 테스트 커버리지 (Phase 4 기준)

모듈	커버리지	비고
전체	55%	주요 로직 위주 테스트
util.py (Celery Tasks)	80%+	자동화 태스크
parallel_processor.py	100%	병렬 처리
classification_service.py	89%	핵심 로직

6. 🚀 설치 및 실행

6.1 사전 요구사항

- **Python:** 3.11+
- **Node.js:** 18+
- **OpenAI API Key:** platform.openai.com
- **Redis Server:** 6.2+ (Celery 브로커용)

6.2 설치 방법

```

# 1. 저장소 클론
git clone https://github.com/jjaayy2222/flownote-mvp.git
cd flownote-mvp

```

```
# 2. 가상환경 생성 및 활성화
python -m venv venv
source venv/bin/activate

# 3. Redis 설치 (macOS)
brew install redis
brew services start redis

# 4. 패키지 설치
pip install -r requirements.txt

# 5. Frontend 패키지 설치
cd web_ui
npm install
cd ..

# 6. 환경변수 설정
cp .env.example .env
# .env 파일에 OpenAI API 키 및 REDIS_URL 설정
```

6.3 전체 서비스 실행 가이드

FlowNote의 모든 기능을 사용하려면 **4개의 터미널** (+ MCP 서버 사용 시 1개 추가)이 필요합니다.

1. FastAPI Backend 실행 (Terminal 1)

```
# 프로젝트 루트 디렉토리에서
# Unix/macOS:
source venv/bin/activate # 또는 pyenv activate <your-env>
# Windows:
# venv\Scripts\activate

python -m uvicorn backend.main:app --reload
# → http://127.0.0.1:8000
```

2. Next.js Frontend 실행 (Terminal 2)

```
cd web_ui
npm run dev
# → http://localhost:3000
# 한국어: http://localhost:3000/ko
# 영어: http://localhost:3000/en
```

3. Celery Worker & Beat 실행 (Terminal 3)

```
# Worker와 Beat 동시 실행 (개발용)
celery -A backend.celery_app.celery worker --beat --loglevel=info
```

4. Flower 모니터링 실행 (Terminal 4)

```
celery -A backend.celery_app.celery flower --port=5555  
# → http://localhost:5555
```

5. MCP 서버 실행 (Terminal 5) - Optional

```
# Claude Desktop 연동 시  
python -m backend.mcp.server  
# 또는 Claude Desktop 설정에서 자동 시작
```

7. 사용 방법

7.1 Step 1: 온보딩

1. Tab 1: 온보딩 이동
2. 이름과 직업 입력 후 영역 선택

7.2 Step 2: 파일 분류

1. Tab 2: 파일 분류 이동
2. 파일 업로드 및 분류 시작

7.3 Step 3: 실시간 모니터링 (v6.0)

1. Dashboard의 Sync Monitor에서 실시간 동기화 상태 확인
2. WebSocket 연결 상태 및 이벤트 로그 모니터링

7.4 Step 4: 충돌 해결 (v6.0)

1. 충돌 발생 시 알림 수신
2. Diff Viewer에서 변경사항 비교
3. Keep Local / Keep Remote / Keep Both 중 선택

7.5 Step 5: 언어 전환 (v6.0)

1. 우측 상단 언어 스위처 클릭
2. 한국어/English 선택
3. URL 자동 변경 및 UI 즉시 업데이트

7.6 Step 6: 자동화 모니터링

1. Flower 대시보드 접속
2. Tasks 탭에서 자동 재분류/리포트 생성 작업 확인
3. System 탭에서 워커 상태 확인

7.7 Step 7: CLI 사용

```
# 단일 파일 분류
python -m backend.cli classify "path/to/file.txt" [user_id]
```

8. 개발 히스토리

Issue별 개발 진행도

Issue	완료일	핵심 기능	상태
[#1-10]	~11/11	Phase 1-2 (MVP)	✓
[#10.4]	12/16	Celery 자동화 & 스케줄링	✓
[#10.11]	02/04	v6.0 Phase 3 (i18n)	✓

주요 커밋 히스토리

- **v5.0** - MCP 서버, Next.js 대시보드, Graph View
- **v6.0 Phase 1** - WebSocket 실시간 업데이트
- **v6.0 Phase 2** - Conflict Diff Viewer
- **v6.0 Phase 3** - 다국어 지원 (i18n) ✓

9. 로드맵

✓ 완료된 기능 (v5.0)

- ✓ 스마트 온보딩 & PARA 분류
- ✓ Celery 기반 비동기 작업 큐
- ✓ 정기 자동 재분류 및 아카이빙 스케줄러
- ✓ Flower 모니터링 통합
- ✓ MCP 서버 구현 ✨
- ✓ Obsidian 동기화 및 충돌 해결 ✨
- ✓ Next.js 기반 모던 대시보드 ✨
- ✓ PARA Graph View & Advanced Stats ✨
- ✓ Mobile Responsive UI ✨

✓ 완료된 기능 (v6.0)

- ✓ **Phase 1: WebSocket** 실시간 업데이트 ✨
 - ✓ Polling 방식 제거
 - ✓ 양방향 실시간 통신
 - ✓ 자동 재연결 메커니즘
 - ✓ 이벤트 기반 UI 업데이트
 - ✓ 네트워크 트래픽 50% 감소

- **Phase 2: Conflict Diff Viewer ✨**

- Monaco Editor 기반 Side-by-Side 비교
- 3가지 해결 옵션 (Keep Local/Remote/Both)
- Markdown 프리뷰
- Syntax Highlighting
- 인라인 Diff 표시

- **Phase 3: 다국어 지원 (i18n) ✨**

- next-intl 기반 다국어 시스템
- 한국어/영어 완벽 지원
- URL 기반 언어 라우팅
- Backend API 응답 현지화
- SEO 메타데이터 다국어화
- 날짜/숫자 로케일별 포맷팅

진행 예정 (v7.0)

- 추가 언어 지원 (일본어, 중국어)
 - AI 기반 자동 번역
 - 고급 검색 필터
 - 파일 버전 히스토리
-

10. ? FAQ

Q1. Redis가 꼭 필요한가요?

A: 네, Celery의 메시지 브로커로 Redis를 사용하므로 반드시 실행되어 있어야 합니다.

Q2. 자동화 작업은 언제 실행되나요?

A: `backend/celery_app/config.py`에 정의된 스케줄에 따릅니다. (예: 재분류-매일 00:00)

11. 🤝 기여하기

Issues 제보 및 PR 환영합니다.

12. 📄 라이선스

MIT License

13. 💬 개발자

Jay (@jjaayy2222)

FlowNote - AI가 당신의 문서를 정리해줍니다 🚀