**요구사항 명세서 v0.3 (대학교 수준)**

**(Software Requirement Specification)**

**개인화된 AI 뉴스 에이전트 시스템 개발**

**헌빈솔호**  
**업체명: 그리드원**  
**지도교수: 이정일**

**📋 목차**

1. **서론**
   1. 서문
   2. 범위
   3. 용어 정의 및 약어
2. **본론**
   1. 제품 개요
   2. 제품 기능
   3. 사용자 클래스 및 특성
   4. 운영 환경
   5. 설계 및 구현 제한 조건
3. **기능적 요구사항**
   1. 시스템 구조
   2. 메인 모듈
   3. 시스템 모듈
   4. 뉴스 데이터 분석 모듈
   5. 외부 API 접근 모듈
   6. 인터페이스 모듈
4. **비기능적 요구사항**
   1. 품질
   2. 운영 환경
   3. 유지보수 및 관리
   4. 테스트 및 검증
5. **인터페이스 요구사항**
   1. 사용자 인터페이스
   2. 시스템 인터페이스
   3. 외부 시스템 인터페이스
6. **부록**
   1. 참고 문서
   2. 변경 이력

**1. 서론**

**1.1 서문**

**1.1.1 개요**

본 문서는 **대학교 수준의 개인화된 AI 뉴스 에이전트 시스템**의 요구사항을 명세합니다. 파이썬 기반의 객체지향 설계를 학습하고 실습할 수 있도록 **적절한 복잡도**를 유지하면서도 실제 동작하는 시스템 구현을 목표로 합니다.

**1.2 범위**

**1.2.1 주요 대상 및 목표**

**학습 목표:**

* 객체지향 프로그래밍의 기본 원칙 이해 및 적용
* 외부 API 연동 및 데이터 처리 경험
* 웹 기반 사용자 인터페이스 구현
* 파이썬 생태계의 주요 라이브러리 활용

**시스템 목표:**

* 사용자의 관심사에 맞춘 뉴스 요약 제공
* 간단한 AI 기반 텍스트 처리
* 다양한 디바이스에서 접근 가능한 웹 인터페이스

**1.2.2 시스템 경계**

**포함(In Scope):**

* 기본적인 사용자 프로파일 관리
* 뉴스 API를 통한 데이터 수집
* OpenAI API를 활용한 요약 기능
* 웹 기반 사용자 인터페이스

**제외(Out of Scope):**

* 복잡한 사용자 인증 시스템
* 대용량 데이터 처리 및 분산 시스템
* 고급 보안 기능
* 상용 서비스 수준의 확장성

**1.3 용어 정의 및 약어**

| **용어/약어** | **정의** |
| --- | --- |
| **클래스** | 객체지향 프로그래밍의 기본 단위 |
| **모듈** | 관련 기능을 그룹화한 파이썬 파일 |
| **API** | 외부 서비스와 통신하기 위한 인터페이스 |
| **JSON** | 데이터 교환을 위한 텍스트 형식 |
| **쿠키** | 웹 브라우저에 저장되는 사용자 정보 |

**2. 본론**

**2.1 제품 이름**

**개인화된 AI 뉴스 에이전트 시스템 (대학교 버전)**

**2.2 제품 개요**

본 시스템은 **대학교 프로젝트에 적합한 규모**로 설계된 AI 뉴스 에이전트입니다:

* **학습 중심 설계**: 객체지향 개념을 실습할 수 있는 클래스 구조
* **단순화된 아키텍처**: 이해하기 쉬운 모듈 구조
* **실용적 기능**: 실제 동작하는 뉴스 요약 서비스
* **확장 가능성**: 향후 개선이 가능한 기본 구조

**2.3 제품 기능**

**2.3.1 핵심 기능**

* **사용자 프로파일 관리**: 간단한 관심사 저장 및 관리
* **뉴스 수집**: 뉴스 API를 통한 최신 뉴스 가져오기
* **AI 요약**: OpenAI API를 활용한 뉴스 요약
* **웹 인터페이스**: 사용자 친화적인 웹 페이지

**2.3.2 부가 기능**

* **디바이스 대응**: 모바일, 태블릿, PC 화면 최적화
* **기본 캐싱**: 불필요한 API 호출 방지
* **에러 처리**: 기본적인 예외 상황 대응

**2.4 사용자 클래스 및 특성**

**2.4.1 대상 사용자**

* **학생 및 교수**: 시스템 학습 및 평가를 위한 사용자
* **일반 사용자**: 뉴스 요약 서비스를 원하는 사용자

**2.5 운영 환경**

**2.5.1 개발 환경 (단순화)**

* **언어**: Python 3.8+ (대학에서 일반적으로 사용)
* **웹 프레임워크**: Flask (학습하기 쉬운 경량 프레임워크)
* **데이터 저장**: JSON 파일 (간단한 파일 기반 저장)
* **외부 API**: OpenAI API, News API

**2.5.2 클라이언트 환경**

* **웹 브라우저**: Chrome, Firefox, Safari (최신 버전)
* **디바이스**: 스마트폰, 태블릿, PC

**2.6 설계 및 구현 제한 조건**

**2.6.1 대학교 수준 제한사항**

* **복잡도 제한**: 이해 가능한 수준의 클래스 구조
* **기술 제한**: 대학에서 학습한 기본 기술 활용
* **시간 제약**: 한 학기 내 완성 가능한 범위
* **자원 제약**: 무료 또는 저비용 서비스 활용

**3. 기능적 요구사항**

**3.1 시스템 구조 (단순화)**

NewsAgent (메인 클래스)

├── UserManager (사용자 관리)

├── NewsCollector (뉴스 수집)

├── NewsSummarizer (뉴스 요약)

└── WebInterface (웹 인터페이스)

**3.2 메인 모듈**

**3.2.1 애플리케이션 시작 (SRS\_FUN\_001)**

| **구분** | **내용** |
| --- | --- |
| **요구사항 ID** | SRS\_FUN\_001 |
| **요구사항명** | 시스템 초기화 |
| **구현 클래스** | NewsAgent.\_\_init\_\_() |
| **기능** | Flask 앱 시작, 설정 로드 |
| **난이도** | ⭐⭐ (초급) |

**3.2.2 사용자 프로파일 확인 (SRS\_FUN\_002)**

| **구분** | **내용** |
| --- | --- |
| **요구사항 ID** | SRS\_FUN\_002 |
| **구현 클래스** | UserManager.load\_profile() |
| **저장 방식** | JSON 파일 |
| **학습 요소** | 파일 I/O, JSON 처리 |

**3.3 시스템 모듈**

**3.3.1 사용자 관리 (SRS\_FUN\_004-007)**

class UserManager:

def \_\_init\_\_(self):

self.profiles\_file = "user\_profiles.json"

def load\_profile(self, user\_id: str) -> dict:

"""사용자 프로파일 로드"""

# JSON 파일에서 사용자 정보 읽기

pass

def create\_profile(self, user\_id: str, interests: list) -> dict:

"""새 사용자 프로파일 생성"""

# 기본 프로파일 생성 및 저장

pass

def update\_profile(self, user\_id: str, new\_interests: list):

"""프로파일 업데이트"""

# 관심사 정보 업데이트

pass

**3.4 뉴스 데이터 분석 모듈**

**3.4.1 뉴스 수집 (SRS\_FUN\_008-009)**

class NewsCollector:

def \_\_init\_\_(self, api\_key: str):

self.api\_key = api\_key

self.base\_url = "https://newsapi.org/v2/"

def fetch\_news(self, category: str, limit: int = 10) -> list:

"""뉴스 API에서 뉴스 가져오기"""

# requests 라이브러리를 사용한 API 호출

pass

def filter\_by\_interests(self, news\_list: list, interests: list) -> list:

"""관심사에 따른 뉴스 필터링"""

# 간단한 키워드 매칭

pass

**3.4.2 뉴스 요약 (SRS\_FUN\_010-011)**

class NewsSummarizer:

def \_\_init\_\_(self, openai\_key: str):

self.openai\_key = openai\_key

def summarize\_articles(self, articles: list) -> str:

"""OpenAI API를 사용한 뉴스 요약"""

# 간단한 프롬프트로 요약 생성

pass

def create\_personalized\_summary(self, articles: list, interests: list) -> str:

"""개인화된 요약 생성"""

# 관심사를 반영한 요약

pass

**3.5 외부 API 연동**

**3.5.1 API 관리 (단순화)**

class APIManager:

def \_\_init\_\_(self):

self.news\_api = NewsCollector(news\_api\_key)

self.ai\_api = NewsSummarizer(openai\_key)

def call\_news\_api(self, params: dict) -> dict:

"""뉴스 API 호출"""

# 기본적인 API 호출 및 에러 처리

pass

def call\_openai\_api(self, prompt: str) -> str:

"""OpenAI API 호출"""

# 간단한 요약 요청

pass

**3.6 웹 인터페이스**

**3.6.1 Flask 기반 웹 서비스**

from flask import Flask, render\_template, request

class WebInterface:

def \_\_init\_\_(self, news\_agent):

self.app = Flask(\_\_name\_\_)

self.news\_agent = news\_agent

self.setup\_routes()

def setup\_routes(self):

@self.app.route('/')

def home():

return render\_template('index.html')

@self.app.route('/summary')

def get\_summary():

# 뉴스 요약 생성 및 반환

pass

**4. 비기능적 요구사항 (대학교 수준)**

**4.1 품질**

**4.1.1 성능 (SRS\_NFN\_001)**

| **구분** | **내용** |
| --- | --- |
| **목표** | 10초 이내 응답 (관대한 기준) |
| **이유** | 학습 목적, 무료 API 제한 고려 |

**4.1.2 정확성 (SRS\_NFN\_002)**

| **구분** | **내용** |
| --- | --- |
| **목표** | 기본적인 요약 품질 확보 |
| **검증** | 수동 검토 및 간단한 테스트 |

**4.2 개발 환경**

**4.2.1 필요 라이브러리 (최소한)**

# requirements.txt (단순화)

flask==2.3.0

requests==2.31.0

openai==0.28.0

python-dotenv==1.0.0

**4.3 테스트 (기본 수준)**

**4.3.1 단위 테스트**

import unittest

class TestNewsAgent(unittest.TestCase):

def test\_profile\_creation(self):

"""프로파일 생성 테스트"""

pass

def test\_news\_fetching(self):

"""뉴스 가져오기 테스트"""

pass

**5. 구현 가이드 (대학교 수준)**

**5.1 개발 단계**

**5.1.1 1단계: 기본 구조 (2주)**

* 클래스 설계 및 기본 구조 구현
* JSON 파일 기반 데이터 저장

**5.1.2 2단계: API 연동 (2주)**

* News API 연동
* OpenAI API 연동

**5.1.3 3단계: 웹 인터페이스 (2주)**

* Flask 기반 웹 페이지
* 기본 HTML/CSS

**5.1.4 4단계: 완성 및 테스트 (1주)**

* 전체 기능 통합
* 간단한 테스트

**5.2 평가 기준**

| **항목** | **배점** | **평가 내용** |
| --- | --- | --- |
| **클래스 설계** | 30% | 객체지향 원칙 적용 |
| **기능 구현** | 40% | 요구사항 충족도 |
| **코드 품질** | 20% | 가독성, 주석 |
| **창의성** | 10% | 추가 기능, 개선사항 |

**5.3 참고 자료**

**5.3.1 학습 자료**

* **Python 공식 문서**: 기본 문법 및 라이브러리
* **Flask 튜토리얼**: 웹 프레임워크 기초
* **API 문서**: News API, OpenAI API 사용법

**5.3.2 예제 코드 구조**

news\_agent\_project/

├── main.py # 메인 실행 파일

├── models/ # 클래스 정의

│ ├── user\_manager.py

│ ├── news\_collector.py

│ └── news\_summarizer.py

├── templates/ # HTML 템플릿

│ └── index.html

├── static/ # CSS, JS 파일

├── data/ # JSON 데이터 파일

├── tests/ # 테스트 파일

└── requirements.txt # 의존성 라이브러리

**6. 부록**

**6.1 기술 스택 (대학교 적합)**

| **구분** | **기술** | **이유** |
| --- | --- | --- |
| **언어** | Python 3.8+ | 학습하기 쉬운 언어 |
| **웹 프레임워크** | Flask | 경량, 단순함 |
| **데이터 저장** | JSON 파일 | 별도 DB 불필요 |
| **외부 API** | News API, OpenAI | 무료 플랜 제공 |

**6.2 예상 학습 효과**

1. **객체지향 프로그래밍**: 클래스와 모듈의 실제 활용
2. **API 연동**: REST API 호출 및 JSON 처리
3. **웹 개발**: Flask를 통한 웹 서비스 구현
4. **프로젝트 관리**: 단계별 개발 진행

**6.3 주의사항**

**6.3.1 API 사용 제한**

* **News API**: 무료 플랜 일일 1000건 제한
* **OpenAI API**: 사용량 기반 과금 (소액)

**6.3.2 데이터 보안**

* API 키는 환경변수로 관리
* 개인정보는 최소한으로 수집

**6.4 변경 이력**

| **버전** | **일자** | **변경 내역** |
| --- | --- | --- |
| 0.3 | 2025-06-21 | **대학교 수준으로 단순화 및 조정** |

**🎓 결론**

이 요구사항 명세서는 **대학교 수준의 프로젝트**에 적합하도록 다음과 같이 조정되었습니다:

**핵심 조정사항**

1. **복잡도 감소**: 고급 아키텍처 패턴 제거, 기본 객체지향 개념 중심
2. **기술 스택 단순화**: Flask, JSON 파일 등 학습하기 쉬운 기술 선택
3. **학습 목표 명확화**: 객체지향 프로그래밍 실습에 중점
4. **현실적 범위**: 한 학기 내 완성 가능한 적절한 분량
5. **평가 기준 제시**: 대학 과제로서의 평가 항목 명시

**교육적 가치**

* ✅ **객체지향 설계 실습**
* ✅ **외부 API 연동 경험**
* ✅ **웹 서비스 구현 기초**
* ✅ **실제 동작하는 프로그램 완성**

이제 대학교 프로젝트로서 적절한 수준의 도전적이면서도 완성 가능한 요구사항이 되었습니다! 🎯 ├── NewsAnalyzer │ └── SummaryCache ├── InterfaceModule (인터페이스) │ └── DeviceRenderer (추상 클래스) │ ├── SmartphoneRenderer │ ├── TabletRenderer │ └── PCRenderer └── ExternalAPI (외부 연동) ├── NewsAPIInterface ├── OpenAIInterface └── CloudStorageInterface

### 3.2 메인 모듈 (NewsAgentMain)

#### 3.2.1 애플리케이션 시작 (SRS\_FUN\_001)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_001 |

| \*\*요구사항명\*\* | 비동기 애플리케이션 시작 |

| \*\*개요\*\* | FastAPI 기반 비동기 웹 애플리케이션 초기화 |

| \*\*구현 클래스\*\* | `NewsAgentMain.start\_application()` |

| \*\*시나리오\*\* | (1) 의존성 주입 컨테이너 초기화<br>(2) 외부 API 연결 검증<br>(3) 캐시 시스템 초기화 |

| \*\*성능 요구사항\*\* | 초기화 시간 3초 이내 |

#### 3.2.2 사용자 프로파일 확인 (SRS\_FUN\_002)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_002 |

| \*\*요구사항명\*\* | 비동기 프로파일 조회 |

| \*\*개요\*\* | MongoDB에서 사용자 프로파일 존재 여부 확인 |

| \*\*구현 클래스\*\* | `NewsAgentMain.check\_user\_profile()` |

| \*\*입력\*\* | 사용자 식별자 (문자열) |

| \*\*출력\*\* | `bool` (프로파일 존재 여부) |

| \*\*데이터 형식\*\* | JSON 직렬화된 `UserProfile` 데이터클래스 |

#### 3.2.3 쿠키 권한 요청 (SRS\_FUN\_003)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_003 |

| \*\*요구사항명\*\* | GDPR 준수 쿠키 동의 |

| \*\*개요\*\* | 웹 표준을 준수한 쿠키 사용 동의 처리 |

| \*\*구현 클래스\*\* | `NewsAgentMain.request\_cookie\_permission()` |

| \*\*준수 규정\*\* | GDPR, 개인정보보호법, 쿠키 정책 |

### 3.3 시스템 모듈 (SystemModule)

#### 3.3.1 쿠키 데이터 접근 (SRS\_FUN\_004)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_004 |

| \*\*요구사항명\*\* | 안전한 쿠키 데이터 접근 |

| \*\*구현 클래스\*\* | `SystemModule.access\_cookie\_data()` |

| \*\*반환 타입\*\* | `Optional[Dict[str, Any]]` |

| \*\*보안 요구사항\*\* | 사용자 동의 검증, 데이터 암호화 |

#### 3.3.2 개인화 프로파일 생성 (SRS\_FUN\_005)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_005 |

| \*\*요구사항명\*\* | ML 기반 프로파일 생성 |

| \*\*구현 클래스\*\* | `ProfileManager.create\_personalized\_profile()` |

| \*\*입력 타입\*\* | `Dict[str, Any]` (쿠키 데이터) |

| \*\*출력 타입\*\* | `UserProfile` (데이터클래스) |

| \*\*알고리즘\*\* | 쿠키 패턴 분석 → 관심사 추출 → 선호도 계산 |

#### 3.3.3 디폴트 프로파일 생성 (SRS\_FUN\_006)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_006 |

| \*\*요구사항명\*\* | 기본 프로파일 생성 |

| \*\*구현 클래스\*\* | `ProfileManager.create\_default\_profile()` |

| \*\*기본 카테고리\*\* | `NewsCategory.POLITICS`, `ECONOMY`, `SOCIETY` |

| \*\*확장성\*\* | 향후 기계학습 기반 개선 가능 |

#### 3.3.4 프로파일 주기적 업데이트 (SRS\_FUN\_007)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_007 |

| \*\*요구사항명\*\* | 자동 프로파일 갱신 |

| \*\*구현 클래스\*\* | `ProfileManager.update\_profile\_if\_needed()` |

| \*\*업데이트 주기\*\* | 7일 (timedelta 객체로 관리) |

| \*\*트리거\*\* | 사용자 접속 시 자동 검사 |

### 3.4 뉴스 데이터 분석 모듈 (NewsAnalysisModule)

#### 3.4.1 외부 API 호출 (SRS\_FUN\_008)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_008 |

| \*\*요구사항명\*\* | 비동기 API 관리 |

| \*\*구현 클래스\*\* | `NewsAnalysisModule.call\_external\_api()` |

| \*\*에러 처리\*\* | 재시도 메커니즘, 대체 API 지원 |

| \*\*모니터링\*\* | API 응답 시간, 성공률 추적 |

#### 3.4.2 뉴스 데이터 크롤링 (SRS\_FUN\_009)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_009 |

| \*\*요구사항명\*\* | 비동기 뉴스 수집 |

| \*\*구현 클래스\*\* | `NewsCrawler.crawl\_news\_data()` |

| \*\*입력 타입\*\* | `List[NewsCategory]` |

| \*\*출력 타입\*\* | `List[NewsData]` |

| \*\*최소 소스\*\* | 3개 이상의 신뢰할 수 있는 뉴스 소스 |

#### 3.4.3 AI 기반 뉴스 요약 (SRS\_FUN\_010)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_010 |

| \*\*요구사항명\*\* | 개인화 AI 요약 |

| \*\*구현 클래스\*\* | `NewsAnalyzer.summarize\_with\_profile()` |

| \*\*입력\*\* | `List[NewsData]`, `UserProfile` |

| \*\*출력\*\* | `str` (500자 이내 요약) |

| \*\*AI 모델\*\* | OpenAI GPT-4 (토큰 최적화) |

#### 3.4.4 AI 재질의 및 검증 (SRS\_FUN\_011)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_011 |

| \*\*요구사항명\*\* | 이중 검증 시스템 |

| \*\*구현 클래스\*\* | `NewsAnalyzer.verify\_and\_requery()` |

| \*\*검증 방식\*\* | 요약 내용 기반 사실 확인 질의 |

| \*\*정확도 목표\*\* | 90% 이상 |

#### 3.4.5 뉴스 요약 갱신 (SRS\_FUN\_012)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_012 |

| \*\*요구사항명\*\* | 스마트 캐시 관리 |

| \*\*구현 클래스\*\* | `SummaryCache.get\_valid\_summary()` |

| \*\*유효 시간\*\* | 3시간 (datetime 객체로 관리) |

| \*\*갱신 전략\*\* | 사용자 접근 시 또는 백그라운드 갱신 |

### 3.5 외부 API 접근 모듈

#### 3.5.1 뉴스 API 연동 (SRS\_FUN\_013)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_013 |

| \*\*요구사항명\*\* | RESTful 뉴스 API 연동 |

| \*\*구현 클래스\*\* | `NewsAPIInterface` |

| \*\*상속 관계\*\* | `APIInterface` 추상 클래스 상속 |

| \*\*인증 방식\*\* | API 키 기반 인증 |

#### 3.5.2 OpenAI API 연동 (SRS\_FUN\_014)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_014 |

| \*\*요구사항명\*\* | AI 모델 API 연동 |

| \*\*구현 클래스\*\* | `OpenAIInterface` |

| \*\*모델\*\* | GPT-4 (최신 버전) |

| \*\*토큰 관리\*\* | 비용 최적화를 위한 토큰 사용량 추적 |

### 3.6 인터페이스 모듈 (InterfaceModule)

#### 3.6.1 멀티 디바이스 출력 (SRS\_FUN\_015)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_FUN\_015 |

| \*\*요구사항명\*\* | 전략 패턴 기반 렌더링 |

| \*\*구현 클래스\*\* | `InterfaceModule.render\_for\_device()` |

| \*\*디바이스 감지\*\* | User-Agent 문자열 분석 |

| \*\*렌더러\*\* | `SmartphoneRenderer`, `TabletRenderer`, `PCRenderer` |

---

## 4. 비기능적 요구사항

### 4.1 품질

#### 4.1.1 성능 (SRS\_NFN\_001)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_NFN\_001 |

| \*\*목표 응답시간\*\* | 5초 이내 (전체 처리) |

| \*\*세부 목표\*\* | 초기 로딩 3초, 요약 생성 5초, API 응답 2초 |

| \*\*모니터링\*\* | `PerformanceMonitor` 클래스로 실시간 추적 |

#### 4.1.2 정확성 (SRS\_NFN\_002)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_NFN\_002 |

| \*\*정확도 목표\*\* | 90% 이상 |

| \*\*검증 방법\*\* | 이중 질의 메커니즘, 사용자 피드백 |

#### 4.1.3 가용성 (SRS\_NFN\_003)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_NFN\_003 |

| \*\*가용성 목표\*\* | 99% 이상 |

| \*\*장애 대응\*\* | API 대체 로직, 자동 재시도 |

#### 4.1.4 확장성 (SRS\_NFN\_004)

| 구분 | 내용 |

|------|------|

| \*\*요구사항 ID\*\* | SRS\_NFN\_004 |

| \*\*아키텍처\*\* | 마이크로서비스 준비 설계 |

| \*\*확장 방향\*\* | 수평적 확장 가능 구조 |

### 4.2 운영 환경

#### 4.2.1 브라우저 호환성 (SRS\_NFN\_005)

| 브라우저 | 최소 버전 | 지원 기능 |

|----------|-----------|-----------|

| Chrome | 90+ | 전체 기능 |

| Firefox | 88+ | 전체 기능 |

| Safari | 14+ | 전체 기능 |

#### 4.2.2 반응형 디자인 (SRS\_NFN\_006)

| 구분 | 내용 |

|------|------|

| \*\*설계 원칙\*\* | 모바일 퍼스트 |

| \*\*브레이크포인트\*\* | 320px, 768px, 1024px, 1200px |

| \*\*구현 방식\*\* | CSS Grid, Flexbox |

### 4.3 유지보수 및 관리

#### 4.3.1 로그 관리 (SRS\_NFN\_007)

| 구분 | 내용 |

|------|------|

| \*\*구현 클래스\*\* | `Logger` |

| \*\*로그 레벨\*\* | ERROR, WARNING, INFO, DEBUG |

| \*\*보관 기간\*\* | 7일 (자동 로테이션) |

| \*\*개인정보 보호\*\* | 자동 마스킹 처리 |

#### 4.3.2 모니터링 (SRS\_NFN\_008)

| 구분 | 내용 |

|------|------|

| \*\*구현 클래스\*\* | `PerformanceMonitor` |

| \*\*추적 항목\*\* | 응답시간, 에러율, API 사용량 |

| \*\*알림\*\* | 임계값 초과 시 실시간 알림 |

### 4.4 테스트 및 검증

#### 4.4.1 단위 테스트 (SRS\_NFN\_009)

| 구분 | 내용 |

|------|------|

| \*\*프레임워크\*\* | pytest, pytest-asyncio |

| \*\*커버리지 목표\*\* | 80% 이상 |

| \*\*테스트 타입\*\* | 유닛 테스트, 통합 테스트 |

#### 4.4.2 타입 검사 (SRS\_NFN\_010)

| 구분 | 내용 |

|------|------|

| \*\*도구\*\* | mypy, pylint |

| \*\*타입 힌팅\*\* | 전체 코드베이스 100% 적용 |

| \*\*정적 분석\*\* | CI/CD 파이프라인 통합 |

---

## 5. 인터페이스 요구사항

### 5.1 사용자 인터페이스

#### 5.1.1 메인 화면 (SRS\_INF\_001)

| 구분 | 내용 |

|------|------|

| \*\*구현 방식\*\* | FastAPI + Jinja2 템플릿 |

| \*\*상태 관리\*\* | 클라이언트 사이드 JavaScript |

| \*\*실시간 업데이트\*\* | WebSocket 연결 |

#### 5.1.2 쿠키 동의 팝업 (SRS\_INF\_002)

| 구분 | 내용 |

|------|------|

| \*\*준수 표준\*\* | GDPR Article 7 |

| \*\*구현\*\* | 모달 다이얼로그 |

| \*\*저장\*\* | 로컬 스토리지 (동의 기록) |

### 5.2 시스템 인터페이스

#### 5.2.1 데이터 모델 인터페이스 (SRS\_INF\_004)

| 구분 | 내용 |

|------|------|

| \*\*구현\*\* | Pydantic 모델 + 데이터클래스 |

| \*\*직렬화\*\* | JSON (사용자 프로파일) |

| \*\*검증\*\* | 자동 타입 검증 및 변환 |

#### 5.2.2 로깅 인터페이스 (SRS\_INF\_005)

| 구분 | 내용 |

|------|------|

| \*\*포맷\*\* | 구조화된 JSON 로그 |

| \*\*출력\*\* | 파일 + 콘솔 (개발 환경) |

| \*\*로테이션\*\* | 일일 로그 파일 생성 |

### 5.3 외부 시스템 인터페이스

#### 5.3.1 뉴스 API 인터페이스 (SRS\_INF\_006)

| 구분 | 내용 |

|------|------|

| \*\*프로토콜\*\* | HTTP/2, RESTful API |

| \*\*인증\*\* | API 키 헤더 방식 |

| \*\*데이터 형식\*\* | JSON |

| \*\*에러 처리\*\* | HTTP 상태 코드 기반 |

#### 5.3.2 OpenAI API 인터페이스 (SRS\_INF\_007)

| 구분 | 내용 |

|------|------|

| \*\*모델\*\* | gpt-4-turbo (최신) |

| \*\*토큰 관리\*\* | 요청별 토큰 사용량 추적 |

| \*\*스트리밍\*\* | 실시간 응답 스트리밍 지원 |

#### 5.3.3 데이터베이스 인터페이스 (SRS\_INF\_008)

| 구분 | 내용 |

|------|------|

| \*\*데이터베이스\*\* | MongoDB 6.0+ |

| \*\*드라이버\*\* | Motor (비동기 MongoDB 드라이버) |

| \*\*연결 관리\*\* | 커넥션 풀링 |

| \*\*데이터 모델\*\* | Document 기반 스키마 |

---

## 6. 부록

### 6.1 참고 문서

#### 6.1.1 기술 스택

| 구분 | 기술 | 버전 | 용도 |

|------|------|------|------|

| \*\*언어\*\* | Python | 3.11+ | 주 개발 언어 |

| \*\*웹 프레임워크\*\* | FastAPI | 0.104+ | 비동기 웹 API |

| \*\*데이터베이스\*\* | MongoDB | 6.0+ | 사용자 프로파일 저장 |

| \*\*캐시\*\* | Redis | 7.0+ | 세션 및 캐시 관리 |

| \*\*AI 서비스\*\* | OpenAI API | GPT-4 | 뉴스 요약 및 분석 |

| \*\*테스트\*\* | pytest | 7.0+ | 단위 및 통합 테스트 |

| \*\*타입 검사\*\* | mypy | 1.5+ | 정적 타입 검사 |

#### 6.1.2 Python 패키지 의존성

```python

# requirements.txt

fastapi>=0.104.0

uvicorn[standard]>=0.23.0

motor>=3.3.0 # MongoDB 비동기 드라이버

redis>=5.0.0

aiohttp>=3.8.0

pydantic>=2.4.0

openai>=1.0.0

pytest>=7.4.0

pytest-asyncio>=0.21.0

mypy>=1.5.0

**6.1.3 데이터 모델 정의**

# 핵심 데이터 모델 (파이썬 데이터클래스)

@dataclass

class UserProfile:

profile\_id: str

interests: List[NewsCategory]

preferences: Dict[str, float]

created\_at: datetime

updated\_at: datetime

is\_default: bool = False

@dataclass

class NewsData:

article\_id: str

title: str

content: str

category: NewsCategory

source: str

published\_at: datetime

url: str

@dataclass

class NewsSummary:

summary\_id: str

original\_news: List[NewsData]

summary\_text: str

user\_profile\_id: str

created\_at: datetime

expires\_at: datetime

verified: bool = False

**6.1.4 지원 뉴스 카테고리**

class NewsCategory(Enum):

POLITICS = "정치"

ECONOMY = "경제"

SOCIETY = "사회"

CULTURE = "문화"

SPORTS = "스포츠"

TECHNOLOGY = "과학기술"

INTERNATIONAL = "국제"

**6.1.5 시스템 설정**

class SystemConfig:

def \_\_init\_\_(self):

self.performance\_config = {

'init\_timeout': 3, # 초기화 시간 제한 (초)

'summary\_timeout': 5, # 요약 생성 시간 제한 (초)

'api\_timeout': 2, # API 응답 시간 제한 (초)

}

self.quality\_config = {

'accuracy\_threshold': 0.9, # 90% 정확성 보장

'availability\_target': 0.99, # 99% 가용성 목표

}

self.cache\_config = {

'summary\_validity\_hours': 3, # 요약 유효 시간

'max\_cache\_size': 1000, # 최대 캐시 크기

}

**6.2 아키텍처 설계 원칙**

**6.2.1 SOLID 원칙 적용**

| **원칙** | **적용 사례** | **구현 클래스** |
| --- | --- | --- |
| **단일 책임** | 각 클래스는 하나의 기능만 담당 | CookieDataAnalyzer, NewsCrawler |
| **개방-폐쇄** | 확장에는 열려있고 수정에는 닫혀있음 | APIInterface 추상 클래스 |
| **리스코프 치환** | 하위 클래스는 상위 클래스를 대체 가능 | DeviceRenderer 상속 구조 |
| **인터페이스 분리** | 클라이언트는 필요한 인터페이스만 의존 | 각 API 인터페이스 분리 |
| **의존성 역전** | 고수준 모듈이 저수준 모듈에 의존하지 않음 | 의존성 주입 패턴 |

**6.2.2 디자인 패턴 활용**

| **패턴** | **용도** | **구현 클래스** |
| --- | --- | --- |
| **팩토리 패턴** | 객체 생성 캡슐화 | APIFactory, NewsAgentFactory |
| **전략 패턴** | 디바이스별 렌더링 알고리즘 | DeviceRenderer 구현체들 |
| **싱글톤 패턴** | 설정 및 로거 관리 | SystemConfig, Logger |
| **옵저버 패턴** | 성능 모니터링 | PerformanceMonitor |
| **캐시 패턴** | 요약 결과 캐싱 | SummaryCache |

**6.3 보안 및 프라이버시**

**6.3.1 데이터 보안 정책**

| **구분** | **내용** | **구현 방법** |
| --- | --- | --- |
| **데이터 암호화** | 사용자 프로파일 AES-256 암호화 | cryptography 라이브러리 |
| **API 키 관리** | 환경 변수로 민감 정보 관리 | python-dotenv |
| **로그 마스킹** | 개인정보 자동 마스킹 | 정규표현식 기반 필터링 |
| **세션 보안** | JWT 토큰 기반 인증 | python-jose |

**6.3.2 법적 준수사항**

* **개인정보보호법** 준수: 쿠키 데이터 처리 시 사용자 동의 필수
* **GDPR** 규정 준수: EU 사용자 대상 데이터 처리 권리 보장
* **저작권법** 준수: 뉴스 콘텐츠 인용 시 출처 명시 및 공정 이용
* **웹 접근성 지침** 준수: WCAG 2.1 AA 수준 준수

**6.4 성능 최적화 전략**

**6.4.1 비동기 처리 최적화**

# 비동기 처리 예시

async def process\_user\_request(self, user\_id: str) -> str:

# 병렬 처리로 성능 향상

profile\_task = asyncio.create\_task(self.get\_user\_profile(user\_id))

news\_task = asyncio.create\_task(self.fetch\_latest\_news())

profile, news\_data = await asyncio.gather(profile\_task, news\_task)

# 요약 생성

summary = await self.generate\_summary(news\_data, profile)

return summary

**6.4.2 캐싱 전략**

| **캐시 레벨** | **대상** | **유효 시간** | **구현** |
| --- | --- | --- | --- |
| **메모리 캐시** | 사용자 프로파일 | 1시간 | Python dict |
| **Redis 캐시** | 뉴스 요약 | 3시간 | Redis TTL |
| **HTTP 캐시** | 정적 리소스 | 24시간 | FastAPI 미들웨어 |

**6.4.3 데이터베이스 최적화**

# MongoDB 인덱스 설정

user\_profiles.create\_index([("profile\_id", 1)], unique=True)

user\_profiles.create\_index([("updated\_at", 1)])

news\_summaries.create\_index([("user\_profile\_id", 1), ("expires\_at", 1)])

**6.5 확장성 고려사항**

**6.5.1 마이크로서비스 전환 준비**

| **서비스** | **책임** | **독립 배포 가능성** |
| --- | --- | --- |
| **User Service** | 사용자 프로파일 관리 | ✅ 높음 |
| **News Service** | 뉴스 수집 및 분석 | ✅ 높음 |
| **AI Service** | AI 요약 및 검증 | ✅ 높음 |
| **Interface Service** | UI 렌더링 | ⚠️ 중간 |

**6.5.2 수평적 확장 전략**

# 로드 밸런싱을 위한 상태 비저장 설계

class NewsAgentMain:

def \_\_init\_\_(self, external\_cache: Redis, external\_db: MongoDB):

# 외부 저장소 의존으로 상태 비저장 보장

self.cache = external\_cache

self.db = external\_db

**6.6 모니터링 및 알림**

**6.6.1 핵심 지표 (KPI)**

| **지표** | **목표값** | **측정 방법** | **알림 임계값** |
| --- | --- | --- | --- |
| **응답 시간** | 5초 이내 | PerformanceMonitor | 8초 초과 |
| **정확도** | 90% 이상 | 사용자 피드백 | 85% 미만 |
| **가용성** | 99% 이상 | 헬스체크 API | 95% 미만 |
| **메모리 사용량** | 80% 이하 | 시스템 모니터링 | 90% 초과 |

**6.6.2 로그 분석 및 대시보드**

# 구조화된 로깅 예시

logger.info(

"News summary generated",

extra={

"user\_id": user\_id,

"summary\_length": len(summary\_text),

"processing\_time\_ms": processing\_time,

"accuracy\_score": accuracy\_score,

"news\_sources": len(news\_sources)

}

)

**6.7 변경 이력**

| **버전** | **일자** | **저자** | **변경 내역** |
| --- | --- | --- | --- |
| 0.1 | 2025-06-20 | 나정호 | 초안 작성 |
| 0.2 | 2025-06-20 | 나정호 | 요구사항 검토 및 수정 |
| **0.3** | **2025-06-21** | **나정호** | **파이썬 클래스 설계 반영, 아키텍처 구체화** |

**6.8 개발 로드맵**

**6.8.1 단계별 개발 계획**

| **단계** | **기간** | **주요 기능** | **완성도** |
| --- | --- | --- | --- |
| **Phase 1** | 2주 | 핵심 클래스 구조, 기본 API 연동 | 40% |
| **Phase 2** | 2주 | 사용자 프로파일, 뉴스 크롤링 | 70% |
| **Phase 3** | 2주 | AI 요약, 캐싱 시스템 | 90% |
| **Phase 4** | 1주 | UI/UX, 성능 최적화 | 100% |

**6.8.2 향후 확장 계획**

* **다국어 지원**: i18n 국제화 프레임워크 도입
* **모바일 앱**: React Native 기반 모바일 애플리케이션
* **고급 AI**: 자체 AI 모델 훈련 및 배포
* **실시간 협업**: 사용자 간 뉴스 공유 및 토론 기능

**6.9 품질 보증**

**6.9.1 테스트 전략**

# 테스트 예시

@pytest.mark.asyncio

async def test\_news\_summary\_generation():

# Given

user\_profile = UserProfile(

profile\_id="test\_user",

interests=[NewsCategory.TECHNOLOGY],

preferences={"technology": 0.9},

created\_at=datetime.now(),

updated\_at=datetime.now()

)

# When

summary = await news\_analyzer.summarize\_with\_profile(

news\_data, user\_profile

)

# Then

assert len(summary) <= 500

assert "기술" in summary or "테크" in summary

**6.9.2 코드 품질 관리**

| **도구** | **용도** | **설정** |
| --- | --- | --- |
| **black** | 코드 포매팅 | line-length: 88 |
| **isort** | import 정렬 | profile: black |
| **flake8** | 린팅 | max-line-length: 88 |
| **mypy** | 타입 검사 | strict mode |
| **bandit** | 보안 검사 | 자동 스캔 |

**🎯 결론**

본 요구사항 명세서 v0.3은 **현대적인 파이썬 아키텍처**를 기반으로 한 확장 가능하고 유지보수가 용이한 뉴스 에이전트 시스템을 정의합니다.

**핵심 개선사항**

1. **객체지향 설계**: SOLID 원칙을 준수한 모듈화된 클래스 구조
2. **비동기 처리**: 고성능 I/O를 위한 asyncio 기반 설계
3. **타입 안전성**: 전체 코드베이스에 타입 힌팅 적용
4. **확장성**: 마이크로서비스 전환이 용이한 구조
5. **성능 최적화**: 캐싱, 병렬 처리, 데이터베이스 최적화

**기대 효과**

* **개발 생산성 향상**: 명확한 클래스 구조와 인터페이스
* **시스템 안정성**: 포괄적인 에러 처리와 모니터링
* **사용자 경험**: 개인화된 고품질 뉴스 요약 서비스
* **확장 가능성**: 미래 요구사항에 대한 유연한 대응

이 명세서는 대학교 수준의 프로젝트로서 적절한 복잡도를 유지하면서도, 실무에서 사용할 수 있는 수준의 완성도를 목표로 합니다.