**미래내일일경험 프로젝트 결과 보고서**

**(주)그리드원 x 한국폴리텍대학 정수캠퍼스**

**nu:speak**

**AI-Agent 기반 개인화 뉴스추천시스템**

**"정보의 바다에서 당신만의 섬을 발견하세요"**

**지도교수님 : 이정일 교수님**

**기업멘토님 : 서양백 본부장님**

**소속 : 한국폴리텍대학교 정수캠퍼스 인공지능소프트웨어학과**

**개발팀 : 김성민, 김정규, 박창진, 이민솔**

**목차**

**1. 프로젝트 개요** ................................................... 3  
1.1 nu:speak 소개  
1.2 개발 배경 및 필요성  
1.3 핵심 기능 및 특징

**2. 요구사항 명세서** ................................................... 6  
2.1 사용자 시나리오 분석  
2.2 기능적 요구사항  
2.3 비기능적 요구사항

**3. 시스템 설계서** ................................................... 11  
3.1 시스템 아키텍처  
3.2 기술 스택 구성  
3.3 서비스 플로우 설계  
3.4 핵심 기능 설계

**4. 구현 결과서** ................................................... 20  
4.1 핵심 기능 구현 결과  
4.2 화면 구현 결과  
4.3 기술적 구현 성과

**5. 프로젝트 완료 및 확장 가능성** ................................................... 26  
5.1 미구현 기능 및 확장 아이디어  
5.2 프로젝트 진행 과정에서 얻은 인사이트

**1. 프로젝트 개요**

**1.1 nu:speak 소개**

**AI-Agent 기반 개인 맞춤형 뉴스 서비스**

nu:speak는 AI-Agent를 통해 사용자에게 실시간 개인 맞춤형 뉴스를 추천하고, 요약/음성/만화 등 다양한 형식으로 콘텐츠를 가공해 제공하는 차세대 뉴스 소비 플랫폼입니다.

**핵심 가치 제안**

**개인화된 뉴스 큐레이션**

* EQ(Emotional Quotient) 슬라이더를 통한 실시간 관심도 조절
* 사용자 행동 패턴 학습을 통한 지능형 추천

**AI 기반 콘텐츠 변환**

* 3줄 요약으로 핵심 정보 빠른 파악
* 4컷 만화로 시각적 이해도 향상
* TTS 음성 변환으로 멀티태스킹 지원

**대화형 AI-Agent**

* 자연어 기반 뉴스 탐색 및 질의응답
* 개인 맥락을 이해하는 지능형 상호작용

**1.2 개발 배경 및 필요성**

**현재 뉴스 소비의 문제점**

**정보 과부하 현상**  
현대인들은 매일 수많은 뉴스를 접하지만, 정작 '나에게 맞는' 뉴스를 찾기는 어려운 상황입니다. 기존 뉴스 서비스들은 단순한 카테고리 분류나 인기도 기반 추천에 의존하여, 개인의 관심사나 상황에 맞는 맞춤형 서비스가 부족합니다.

**뉴스 소비 방식의 한계**

* 긴 기사를 끝까지 읽기 어려운 현대인의 라이프스타일
* 텍스트 중심의 단조로운 정보 전달 방식
* 수동적인 뉴스 소비에서 벗어나지 못하는 구조적 한계

**nu:speak의 해결 방안**

**사용자 중심의 능동적 뉴스 소비**

* 사용자가 직접 뉴스 소비의 방향성을 결정할 수 있는 EQ 슬라이더
* AI가 사용자를 이해하고 대화하듯 뉴스를 추천하는 에이전트 시스템

**효율적이고 다양한 콘텐츠 경험**

* 핵심만 빠르게 파악할 수 있는 AI 요약 기능
* 텍스트를 넘어 음성, 이미지, 영상으로 확장된 멀티모달 콘텐츠

**1.3 핵심 기능 및 특징**

**1.3.1 EQ 슬라이더 기반 개인화**

**실시간 관심도 조절**  
사용자가 IT 70%, 경제 40%, 정치 60% 등으로 카테고리별 관심도를 실시간으로 조절할 수 있어, 오늘은 IT 집중, 내일은 경제 중심으로 즉시 변경 가능합니다.

**동적 개인화 시스템**  
고정된 설정이 아닌 상황과 기분에 따라 유연하게 조절할 수 있는 개인화 시스템을 통해 진정한 맞춤형 뉴스 소비가 가능합니다.

**1.3.2 AI-Agent 기반 대화형 인터페이스**

**자연어 기반 뉴스 탐색**  
"오늘 개발 트렌드 알려줘", "정치권 개편에 대한 야당 입장은?" 등 자연어로 질문하면 AI-Agent가 관련 뉴스를 찾아 대화 형태로 제공합니다.

**맥락 이해 기반 추천**  
단순한 키워드 매칭이 아닌, 사용자의 질문 의도와 맥락을 이해하여 적절한 뉴스와 추가 정보를 제공합니다.

**1.3.3 AI 기반 콘텐츠 생성**

**3줄 요약 시스템**  
긴 뉴스 기사를 AI가 핵심 내용만 3줄로 압축하여, 바쁜 현대인도 빠르게 정보를 파악할 수 있습니다.

**4컷 만화 자동 생성**  
뉴스 내용을 시각적으로 이해하기 쉽도록 DALL-E 기반 4컷 만화로 자동 변환하여, 복잡한 내용도 재미있고 직관적으로 전달합니다.

**TTS 음성 변환**  
운전 중이나 요리 중 등 화면을 볼 수 없는 상황에서도 뉴스를 청취할 수 있도록 자연스러운 한국어 음성으로 변환 제공합니다.

# 2. 사용자 시나리오 및 요구사항 명세서

## 2.1 사용자 시나리오 분석

### 2.1.1 주요 사용자 프로필

**사용자 A: 김민수 (28세, IT 개발자)**

**특성 및 환경**

* 지하철 출퇴근 35분, 점심시간의 제한된 시간 활용
* IT 트렌드에 민감하며 새로운 기술에 대한 빠른 수용 성향
* 실용적이고 간결한 정보를 선호하는 효율성 중시형 사용자

**주요 니즈**

* 제한된 시간 내에 IT 관련 핵심 정보만 빠르게 파악
* 긴 기사 대신 요약된 정보로 트렌드 파악
* 멀티태스킹이 가능한 음성 기반 뉴스 소비

**도출된 요구사항**

* 실시간으로 IT 뉴스 비중을 높일 수 있는 EQ 슬라이더
* 긴 기사를 3줄로 요약하여 핵심 파악이 가능한 AI 요약 기능
* 음성으로 뉴스를 청취할 수 있는 TTS 기능

**사용자 B: 박지훈 (22세, 정치외교학과 학생)**

**특성 및 환경**

* MZ세대로 짧고 시각적인 콘텐츠에 익숙함
* 복잡한 정치 이슈를 쉽고 재미있게 이해하고 싶어함
* SNS를 통한 정보 공유와 토론을 즐기는 디지털 네이티브

**주요 니즈**

* 어려운 정치 뉴스를 쉽고 재미있게 이해할 수 있는 콘텐츠
* 시각적으로 기억하기 쉬운 형태의 정보 제공
* 친구들과 공유하기 좋은 흥미로운 콘텐츠

**도출된 요구사항**

* 복잡한 정치 이슈를 시각적으로 이해하기 쉬운 4컷 만화 기능
* AI 에이전트를 통한 심층적인 질의응답 기능

### 2.1.2 핵심 사용 시나리오

**시나리오 1: 첫 사용자 온보딩 (김민수)**

1. **관심사 설정 단계**
   * 앱 접속 후 관심 카테고리 선택 (IT, 경제, 스포츠 선택)
   * EQ 슬라이더로 초기 관심도 설정 (IT 60%, 경제 30%, 스포츠 10%)
2. **첫 뉴스 경험**
   * 설정한 관심도에 따른 개인화된 뉴스 목록 제공
   * AI 요약으로 각 뉴스의 핵심 내용 미리 확인
3. **실시간 조절 경험**
   * 지하철에서 IT 뉴스에 더 집중하고 싶어 EQ 슬라이더를 IT 70%로 조정
   * 즉시 IT 뉴스 비중이 높아진 새로운 피드 확인

**시나리오 2: 일상적 뉴스 소비 (김민수)**

1. **효율적 정보 습득**
   * 점심시간 3줄 요약으로 주요 뉴스 10개 빠르게 확인
   * 관심 있는 뉴스는 TTS로 들으며 점심 식사
2. **AI 에이전트 활용**
   * "오늘 개발자들이 관심 가질만한 뉴스 있어?"라고 질문
   * AI가 최신 프레임워크, 개발 도구 관련 뉴스 추천

**시나리오 3: 학습 목적 뉴스 탐색 (박지훈)**

1. **재미있는 정치 뉴스 소비**
   * 복잡한 정치 이슈를 4컷 만화로 쉽게 이해
   * 시각적 콘텐츠를 SNS에 공유하며 친구들과 토론
2. **AI 에이전트를 통한 심층 탐색**
   * "최근 정치권 개편 논의에 대해 알려줘"라고 질문
   * AI가 관련 배경과 다양한 관점의 뉴스들을 체계적으로 제공

## 2.2 기능적 요구사항

### 2.2.1 개인화 기능

**FR-01: EQ 슬라이더 관심도 설정**

* 사용자가 카테고리별 관심도를 0-100% 범위에서 실시간 조절
* 설정 변경 시 즉시 뉴스 피드에 반영
* 사용자 설정 데이터의 영구 저장 및 복원

**FR-02: 개인화된 뉴스 추천**

* EQ 설정에 따른 카테고리별 뉴스 필터링
* 가중치 기반 뉴스 정렬 및 표시
* 사용자 행동 패턴 학습을 통한 추천 정확도 향상

**FR-03: 사용자 행동 분석**

* 뉴스 클릭, 읽기 시간, 공유 등 사용자 행동 데이터 수집
* 개인화 알고리즘 개선을 위한 피드백 시스템
* 사용자 선호도 패턴 분석 및 자동 EQ 조정 제안

### 2.2.2 AI 콘텐츠 생성 기능

**FR-04: AI 3줄 요약**

* OpenAI GPT를 활용한 뉴스 자동 요약
* 원문의 핵심 내용을 3문장으로 압축
* 요약 품질 검증 및 일관성 유지

**FR-05: AI 4컷 만화 생성**

* DALL-E를 활용한 뉴스 기반 4컷 만화 자동 생성
* 뉴스 내용을 시각적 스토리로 변환
* 각 컷별 설명 및 전체 스토리 일관성 유지

**FR-06: TTS 음성 변환**

* 고품질 한국어 음성 합성
* 뉴스 원문 및 요약문의 음성 변환
* 재생 속도 조절 및 음성 품질 선택 기능

### 2.2.3 AI-Agent 대화 기능

**FR-07: 자연어 기반 뉴스 검색**

* 사용자의 자연어 질문 이해 및 의도 분석
* 질문에 맞는 관련 뉴스 검색 및 추천
* 대화 맥락을 고려한 연속적인 질의응답

**FR-08: 지능형 추천 시스템**

* 사용자 질문의 숨은 의도 파악
* 연관 뉴스 및 배경 정보 자동 제공
* 다양한 관점의 뉴스 균형있는 추천

**FR-09: 대화 히스토리 관리**

* 사용자와 AI-Agent 간 대화 내역 저장
* 이전 대화 맥락을 활용한 개인화된 응답
* 학습 기능을 통한 응답 품질 지속 개선

### 2.2.4 사용자 인터페이스 기능

**FR-10: 반응형 웹 인터페이스**

* 모바일, 태블릿, 데스크톱 최적화된 UI/UX
* 카테고리별 뉴스 탭 네비게이션
* 뉴스 카드 그리드 레이아웃
* EQ 슬라이더 직관적 인터페이스

**FR-11: 뉴스 데이터 수집 시스템**

* 다양한 뉴스 소스로부터 실시간 뉴스 수집
* 카테고리별 자동 분류 및 태깅
* 중복 뉴스 필터링 및 품질 검증

**FR-12: 멀티미디어 콘텐츠 관리**

* 이미지, 음성, 텍스트 통합 관리
* 콘텐츠 캐싱 및 최적화
* 다양한 디바이스에서 일관된 콘텐츠 제공

## 2.3 비기능적 요구사항

### 2.3.1 사용성 요구사항

* 직관적인 EQ 슬라이더 조작 인터페이스
* 빠른 응답 속도로 실시간 피드백 제공
* 모바일/데스크톱 반응형 지원
* 접근성 가이드라인 준수 (WCAG 2.1)

### 2.3.2 호환성 요구사항

* 모던 웹 브라우저 지원 (Chrome, Firefox, Safari, Edge)
* 모바일 기기 터치 인터페이스 지원
* 다양한 화면 크기에 대한 반응형 디자인
* 크로스 플랫폼 호환성 보장

### 2.3.3 확장성 요구사항

* 사용자 증가에 대응 가능한 시스템 아키텍처
* 새로운 AI 모델 및 API 통합 용이성
* 다국어 지원 확장 가능성
* 추가 뉴스 소스 연동 유연성

### 2.3.4 보안 요구사항

* API 키 및 민감 정보 안전한 관리
* 사용자 데이터 프라이버시 보호
* HTTPS 통신 및 데이터 암호화
* 외부 API 호출 시 보안 검증

# 3. 시스템 설계서

## 3.1 시스템 아키텍처

### 3.1.1 전체 시스템 구조

스크린샷, 도표, 라인, 스케치이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.

nu:speak 시스템은 **4계층 아키텍처**로 설계되어 사용자 경험부터 데이터 처리까지 체계적으로 구성됩니다:

**사용자 레이어 (User Layer)**

* 웹 브라우저를 통한 사용자 인터페이스
* 모바일/데스크톱 반응형 지원
* 실시간 상호작용 및 피드백

**프론트엔드 레이어 (Frontend Layer)**

* React + Next.js 기반 SPA(Single Page Application)
* EQ 슬라이더, 뉴스 피드, AI 콘텐츠 표시
* 사용자 행동 데이터 수집 및 전송

**백엔드 레이어 (Backend Layer)**

* Node.js + Express 기반 RESTful API 서버
* AI 서비스 통합 및 개인화 엔진
* 파일 시스템 기반 데이터 저장 및 관리

**외부 서비스 레이어 (External Services Layer)**

* OpenAI API (GPT-3.5, DALL-E 3)
* Google TTS API
* 네이버 뉴스 크롤링 시스템

### 3.1.2 주요 데이터 플로우텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진 AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**전체 데이터 흐름:**

1. 네이버 뉴스 크롤링을 통한 실시간 뉴스 수집
2. 백엔드에서 카테고리별 분류 및 파일 저장
3. 사용자 EQ 설정에 따른 개인화 필터링
4. AI API를 통한 멀티모달 콘텐츠 생성
5. 프론트엔드에서 통합된 사용자 경험 제공

## 3.2 기술 스택 구성

### 3.2.1 Frontend Technology Stack

**React + Next.js Framework**

* React 18: 컴포넌트 기반 UI 개발, Hooks 활용
* Next.js 13: SSR/SSG 지원, 파일 기반 라우팅
* TypeScript: 정적 타입 검사로 코드 안정성 확보
* CSS Modules: 컴포넌트 스코프 스타일링

**상태 관리 및 통신**

* React Context: 전역 상태 관리 (EQ 설정, 사용자 데이터)
* Fetch API: HTTP 클라이언트로 Backend API 통신
* localStorage: 클라이언트 사이드 설정 영구 저장

**UI/UX 라이브러리**

* 커스텀 슬라이더 컴포넌트: EQ 조절 인터페이스
* 반응형 그리드 시스템: 뉴스 카드 레이아웃
* 모달 및 네비게이션 컴포넌트

### 3.2.2 Backend Technology Stack

**Node.js + Express Framework**

* Node.js 18: 비동기 I/O 처리, 높은 동시성 지원
* Express.js: RESTful API 서버 구축
* JavaScript: 빠른 개발과 프로토타입 구현

**데이터 저장 및 관리**

* 파일 시스템: 뉴스 데이터 JSON 형태 저장
* 메모리 캐싱: 개인화 결과 임시 저장
* 로그 파일: 시스템 동작 및 오류 추적

**미들웨어 및 보안**

* CORS: 크로스 오리진 요청 처리
* 환경 변수: API 키 및 설정 관리
* 에러 핸들링: 안정적인 서비스 운영

### 3.2.3 AI/ML Technology Stack

**OpenAI Integration**

* GPT-3.5 Turbo: 뉴스 요약, 대화형 응답 생성
* DALL-E 3: 뉴스 기반 4컷 만화 이미지 생성
* API 최적화: 토큰 사용량 관리 및 비용 효율성

**Google Cloud AI**

* Google TTS API: 자연스러운 한국어 음성 합성
* 음성 품질 최적화: 발음 정확도 및 자연스러움

**개인화 시스템**

* 가중치 기반 필터링: EQ 설정에 따른 뉴스 선별
* 실시간 개인화: 즉시 반영되는 맞춤형 추천
* 다양성 보장: 균형잡힌 뉴스 제공 알고리즘

### 3.2.4 External APIs & Services

**뉴스 데이터 소스**

* 네이버 뉴스 크롤링: 실시간 한국 뉴스 수집
* 카테고리별 분류: IT, 경제, 사회, 정치, 세계, 생활/문화
* 중복 제거: 동일 뉴스 필터링 시스템

**배포 및 운영**

* 로컬 개발 환경: 빠른 프로토타입 개발
* 향후 클라우드 확장: AWS, Docker 기반 배포 계획

## 3.3 핵심 기능 설계

### 3.3.1 EQ 슬라이더 기반 개인화 시스템

**텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**시스템 설계 개요** EQ 슬라이더는 사용자의 실시간 관심도 변화를 시스템에 즉시 반영하는 핵심 인터페이스로 설계됩니다. 사용자가 카테고리별 가중치를 조절하면 개인화 엔진이 이를 해석하여 맞춤형 뉴스를 제공합니다.

**핵심 설계 원칙**

* **직관성**: 0-100% 범위의 직관적인 슬라이더 조작
* **실시간성**: 변경 즉시 뉴스 피드에 반영되는 반응형 시스템
* **영구성**: localStorage를 통한 사용자 설정 보존
* **복원력**: 설정 손실 시 기본값으로 안전 복원

**개인화 알고리즘 설계**

1. **1차 필터링**: EQ 가중치 0% 카테고리 자동 제외
2. **2차 가중치 적용**: 각 뉴스에 개인화 점수 부여
3. **3차 다양성 보장**: 동일 소스 과도한 집중 방지
4. **4차 시간 가중치**: 최신성과 개인화 점수 균형
5. **최종 정렬**: 하이브리드 스코어링으로 순위 결정

**사용자 온보딩 플로우** 신규 사용자의 원활한 EQ 설정을 위한 단계별 가이드 시스템을 설계하여, 복잡한 개인화 시스템을 쉽게 이해하고 설정할 수 있도록 지원합니다.

### 3.3.2 AI 콘텐츠 생성 시스템

**텍스트, 도표, 스크린샷, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**멀티모달 콘텐츠 아키텍처** 단일 뉴스 원문을 입력으로 받아 3줄 요약, 4컷 만화, TTS 음성의 세 가지 형태로 동시 변환하는 병렬 처리 시스템을 설계합니다.

**AI 콘텐츠 생성 파이프라인**

* **3줄 요약**: GPT-3.5를 활용한 핵심 정보 추출 및 압축
* **4컷 만화**: DALL-E 3 기반 시각적 스토리텔링 자동 생성
* **TTS 음성**: Google TTS API를 통한 자연스러운 한국어 음성 합성

**품질 보장 시스템 설계**

* **요약 품질**: 원문 대비 정보 손실 최소화, 핵심 키워드 보존
* **만화 일관성**: 스토리 연결성, 시각적 통일성 유지
* **음성 자연성**: 발음 정확도, 억양 자연스러움 확인
* **오류 처리**: API 실패 시 대체 콘텐츠 제공 메커니즘

**성능 최적화 설계** 병렬 처리를 통해 전체 생성 시간을 단축하고, 사용자에게 단계별 결과를 점진적으로 제공하여 체감 속도를 향상시키는 시스템을 설계합니다.

### 3.3.3 AI-Agent 대화 시스템

**텍스트, 도표, 라인, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**대화형 AI 시스템 설계** 사용자와 자연어로 소통하며 맞춤형 뉴스를 제공하는 지능형 에이전트 시스템을 설계합니다. 단순한 검색을 넘어 사용자의 의도를 파악하고 맥락을 이해하는 대화형 인터페이스를 구현합니다.

**의도 분석 엔진 설계**

* **자연어 처리**: 사용자 입력의 의미와 의도 파악
* **엔티티 추출**: 인물, 기관, 날짜 등 핵심 정보 식별
* **맥락 이해**: 이전 대화와 사용자 프로필을 고려한 응답
* **질의 확장**: 동의어, 관련어를 통한 검색 범위 확장

**대화 관리 시스템** 세션별 대화 히스토리를 관리하고, 사용자의 개인화 설정과 연동하여 일관성 있는 대화 경험을 제공하는 시스템을 설계합니다.

**응답 생성 전략**

* **정보 제공**: 요청된 뉴스 정보의 정확한 전달
* **추가 제안**: 연관 뉴스 및 후속 질문 제안
* **개인화 반영**: 사용자의 EQ 설정을 고려한 맞춤형 응답

### 3.3.4 확장 가능한 시스템 아키텍처

**텍스트, 도표, 라인, 평면도이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**데이터베이스 연동 설계** 현재의 파일 기반 시스템에서 PostgreSQL, Redis, MongoDB를 활용한 확장 가능한 데이터 아키텍처로의 발전 방향을 설계합니다.

**마이크로서비스 아키텍처 준비**

* **개인화 서비스**: EQ 기반 추천 엔진 독립화
* **AI 콘텐츠 서비스**: 멀티모달 생성 파이프라인 분리
* **대화 서비스**: AI-Agent 시스템 모듈화
* **뉴스 수집 서비스**: 크롤링 및 분류 시스템 독립화

**성능 및 확장성 고려사항**

* **캐싱 전략**: Redis를 활용한 개인화 결과 고속 캐싱
* **로드 밸런싱**: 트래픽 분산을 위한 서버 구성
* **API 설계**: RESTful 원칙 기반 확장 가능한 인터페이스
* **보안 설계**: 인증, 권한 관리 및 데이터 보호 체계

**클라우드 네이티브 설계** AWS 등 클라우드 환경에서의 배포와 운영을 고려한 컨테이너 기반 아키텍처와 CI/CD 파이프라인 설계를 포함합니다.

# 4. 구현 결과서

# 4.1 핵심 기능 구현 결과

## 4.1.1 AI-Agent 시스템 구현

**AI-Agent의 핵심 역할:** nu:speak의 AI-Agent는 뉴스 콘텐츠를 분석하고 사용자에게 최적화된 형태로 변환하는 지능형 시스템입니다. 실제 구현된 aiService.js를 통해 OpenAI API와 Google TTS API를 통합 관리합니다.

**실제 구현된 AI-Agent 통합 서비스:**

|  |
| --- |
| // aiService.js - 실제 구현된 AI-Agent 통합 서비스  const OpenAI = require("openai");  const axios = require('axios');  const OPENAI\_API\_KEY = process.env.OPENAI\_API\_KEY;  const GOOGLE\_TTS\_API\_KEY = process.env.GOOGLE\_TTS\_API\_KEY;  const openai = OPENAI\_API\_KEY && OPENAI\_API\_KEY !== "DUMMY\_KEY"  ? new OpenAI({ apiKey: OPENAI\_API\_KEY })  : null; |

**AI-Agent 환경 관리 시스템:** 실제 구현에서는 API 키가 없는 환경에서도 더미 데이터를 제공하여 개발과 테스트가 가능하도록 설계했습니다.

## 4.1.2 AI-Agent 기반 콘텐츠 생성

### ****AI-Agent 기반 3줄 요약 생성****

실제 구현된 AI-Agent의 요약 생성 시스템:

|  |
| --- |
| async function getSummary(text) {  if (!OPENAI\_API\_KEY || OPENAI\_API\_KEY === "DUMMY\_KEY") {  return "[더미 요약] 요약 기능을 사용하려면 OpenAI API 키가 필요합니다.";  }  const prompt = `다음 뉴스를 3문장으로 요약해줘:\n${text}`;  const response = await openai.chat.completions.create({  model: "gpt-3.5-turbo",  messages: [{ role: "user", content: prompt }],  max\_tokens: 300  });  return response.choices[0].message.content.trim();  } |

### ****AI-Agent 기반 4컷 만화 생성****

**AI-Agent의 창작 과정:** 뉴스 → 스토리 변환 → 영어 번역 → 이미지 생성의 다단계 프로세스

|  |
| --- |
| async function getComicWithImagesAndSummary(text, title = '') {  if (!OPENAI\_API\_KEY) {  throw new Error('OpenAI API 키가 설정되지 않았습니다.');  }    try {  // 1. AI-Agent가 뉴스를 4컷 스토리로 변환  const storyPrompt = `  다음 뉴스를 4컷만화로 만들어주세요:  제목: ${title}  내용: ${text}  각 컷을 간단명료하게 설명해주세요:  1컷: (상황 소개)  2컷: (전개)  3컷: (클라이막스)  4컷: (결말)`;    const storyResponse = await openai.chat.completions.create({  model: "gpt-3.5-turbo",  messages: [{ role: "user", content: storyPrompt }],  max\_tokens: 400,  temperature: 0.7  });    const storyContent = storyResponse.choices[0].message.content.trim();    // 2. AI-Agent가 각 컷 추출 및 정제  const panels = [];  const lines = storyContent.split('\n').filter(line => line.trim());    for (let i = 1; i <= 4; i++) {  const line = lines.find(l => l.includes(`${i}컷`));  if (line) {  let content = line;  content = content.replace(`${i}컷:`, '');  content = content.replace(`${i}컷：`, '');  content = content.replace(`${i}컷`, '');  content = content.replace('(', '').replace(')', '');  content = content.trim();  panels.push(content);  }}    // 3. AI-Agent가 영어 번역 수행 (DALL-E 최적화)  const translatePrompt = `Translate these Korean comic panels to simple English descriptions:  1: ${panels[0]}  2: ${panels[1]}  3: ${panels[2]}  4: ${panels[3]}`;  const translateResponse = await openai.chat.completions.create({  model: "gpt-3.5-turbo",  messages: [{ role: "user", content: translatePrompt }],  max\_tokens: 200  });    const englishPanels = translateResponse.choices[0].message.content.trim();    // 4. AI-Agent가 DALL-E 3로 2x2 4컷만화 생성  const imagePrompt = `2x2 four-panel comic strip without any text. ${englishPanels} Simple cartoon style with Korean/Asian characters, consistent character designs, clear storytelling.`;    const imgRes = await openai.images.generate({  model: "dall-e-3",  prompt: imagePrompt,  n: 1,  size: "1024x1024",  quality: "standard"  });    // 5. AI-Agent가 통합 요약 생성  const summaryPrompt = `다음 기사를 3문장으로 요약해주세요:  제목: ${title}  내용: ${text}`;    const summaryResponse = await openai.chat.completions.create({  model: "gpt-3.5-turbo",  messages: [{ role: "user", content: summaryPrompt }],  max\_tokens: 200  });    const summary = summaryResponse.choices[0].message.content.trim();    return {  comicImage: imgRes.data[0].url,  comicPanels: panels.map((caption, idx) => ({  panelNumber: idx + 1,  caption: `${idx + 1}컷: ${caption}`  })),  summary,  layoutType: '2x2'  };    } catch (error) {  console.error('만화 생성 오류:', error);    // AI-Agent 실패 시 대체 콘텐츠 제공  const fallbackStory = [  "뉴스의 핵심 상황을 보여주는 장면",  "상황이 전개되는 모습",  "중요한 전환점이나 클라이막스",  "최종 결과나 영향을 보여주는 장면"  ];    return {  comicImage: '',  comicPanels: fallbackStory.map((caption, idx) => ({  panelNumber: idx + 1,  caption: `${idx + 1}컷: ${caption}`  })),  summary: '4컷만화 생성 중 오류가 발생했습니다. 잠시 후 다시 시도해주세요.',  error: error.message,  layoutType: '2x2'  };}} |

### ****AI-Agent 기반 TTS 음성 변환****

**Google TTS API 연동을 통한 음성 생성:**

|  |
| --- |
| async function getGoogleTTS(text) {  if (!GOOGLE\_TTS\_API\_KEY || GOOGLE\_TTS\_API\_KEY === 'DUMMY\_KEY') {  return null;  }  const url = `https://texttospeech.googleapis.com/v1/text:synthesize?key=${GOOGLE\_TTS\_API\_KEY}`;  const body = {  input: { text },  voice: { languageCode: 'ko-KR', name: 'ko-KR-Standard-A' },  audioConfig: { audioEncoding: 'MP3' }  };  const response = await axios.post(url, body);  return response.data.audioContent; // base64 string  } |

## 4.1.3 AI-Agent 모듈 통합 및 내보내기

**실제 구현된 AI-Agent 서비스 모듈:**

|  |
| --- |
| module.exports = {  getSummary,  getComic,  getGoogleTTS,  getComicWithImages,  getComicWithImagesAndSummary,  getSingleImageSummary  }; |

## 4.1.3 AI-Agent 연동 개인화 시스템

**AI-Agent의 개인화 처리 과정:** EQ 슬라이더는 단순한 필터가 아닌, AI-Agent가 사용자의 실시간 관심도를 해석하고 맞춤형 콘텐츠를 생성하는 입력 데이터로 활용됩니다.

### ****EQ 슬라이더 구현 (settings.tsx)****

**실시간 가중치 조절 시스템:**

|  |
| --- |
| // 카테고리별 가중치 상태 관리  const [weights, setWeights] = useState<{ [key: string]: number }>(  Object.fromEntries(categories.map(cat => [cat.key, 50]))  );  // AI-Agent가 해석할 수 있는 가중치 변경 처리  const handleChange = (cat: string, value: number) => {  setWeights(w => ({ ...w, [cat]: value }));  };  // AI-Agent 개인화 설정 영구 저장  const handleSave = () => {  localStorage.setItem('nuspeak\_weights', JSON.stringify(weights));  setSaved(true);  setTimeout(() => setSaved(false), 1500);  };  // 클라이언트 사이드에서 AI-Agent 설정 복원  useEffect(() => {  const saved = localStorage.getItem('nuspeak\_weights');  if (saved) {  try {  setWeights(JSON.parse(saved));  } catch {  // 에러 시 기본값 유지  } }},  ); |

### ****AI-Agent 기반 개인화 필터링 (news.tsx)****

**AI-Agent가 사용자 의도를 해석하는 과정:**

|  |
| --- |
| // AI-Agent가 EQ 설정을 분석하여 개인화된 뉴스 카테고리 결정  const fetchNews = async () => {  setLoading(true);  try {  const category = CATEGORIES.find(cat => cat.id === selectedCategory);  let categoryValue = category ? category.value : ['IT', '경제', '사회', '정치', '세계', '생활/문화'];    // AI-Agent가 추천 카테고리에서 가중치 기반 맞춤화 수행  if (selectedCategory === 'recommended' && Object.keys(weights).length > 0) {  const weightedCategories = Object.entries(weights)  .filter(([\_, weight]) => weight > 0) // 관심 없는 카테고리 제외  .map(([cat, \_]) => cat);    if (weightedCategories.length > 0) {  categoryValue = weightedCategories; // AI-Agent가 선별한 카테고리  }  }    // AI-Agent가 결정한 카테고리로 뉴스 API 호출  let url = 'http://localhost:3001/api/news/personalized';  if (categoryValue.length === 1) {  url += `?newsType=${categoryValue[0]}`;  } else if (categoryValue.length > 1) {  url += `?newsType=${categoryValue.join(',')}`;  }    const response = await fetch(url);  const data = await response.json();    const newsArray = Array.isArray(data) ? data : (data ? [data] : []);  setArticles(newsArray.slice(0, 30));  } catch (error) {  console.error('뉴스 가져오기 오류:', error);  setArticles([]);  } finally {  setLoading(false);  }  };  // AI-Agent가 가중치를 적용하여 뉴스 결과 최적화  const applyWeights = (articles: any[], weights: { [key: string]: number }) => {  const totalWeight = Object.values(weights).reduce((sum, weight) => sum + weight, 0);  if (totalWeight === 0) return articles;  // AI-Agent가 카테고리별 뉴스 분류  const categoryArticles: { [key: string]: any[] } = {};  articles.forEach(article => {  if (!categoryArticles[article.category]) {  categoryArticles[article.category] = [];  }  categoryArticles[article.category].push(article);  });  // AI-Agent가 가중치에 따른 뉴스 비율 계산  const weightedArticles: any[] = [];  Object.entries(weights).forEach(([category, weight]) => {  if (weight > 0 && categoryArticles[category]) {  const ratio = weight / totalWeight;  const count = Math.round(articles.length \* ratio);  const selected = categoryArticles[category].slice(0, count);  weightedArticles.push(...selected);  }  });  // AI-Agent가 다양성 보장을 위해 나머지 뉴스 추가  const remainingArticles = articles.filter(article =>  !weightedArticles.some(w => w.id === article.id)  );  weightedArticles.push(...remainingArticles.slice(0, Math.max(0, 12 - weightedArticles.length)));  return weightedArticles.slice(0, 12);  }; |

### ****AI-Agent 기반 뉴스 수집 시스템 (newsCollector.js)****

**AI-Agent가 관리하는 뉴스 크롤링 및 처리:**

|  |
| --- |
| // AI-Agent가 사용하는 뉴스 소스 맵핑  const NAVER\_NEWS\_URLS = {  'IT': 'https://news.naver.com/section/105',  '경제': 'https://news.naver.com/section/101',  '사회': 'https://news.naver.com/section/102',  '생활/문화': 'https://news.naver.com/section/103',  '세계': 'https://news.naver.com/section/104',  '정치': 'https://news.naver.com/section/100'};  // AI-Agent가 수행하는 지능형 뉴스 수집  async function fetchNaverNews(categories = ['IT', '경제', '사회', '정치', '세계', '생활/문화']) {  let allNews = [];    for (const cat of categories) {  const url = NAVER\_NEWS\_URLS[cat];  if (!url) continue;    try {  console.log(`AI-Agent 크롤링 시도: ${cat} - ${url}`);  const { data } = await axios.get(url, {  headers: {  'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36'  },  timeout: 10000});    const $ = cheerio.load(data);  const articles = [];    // AI-Agent가 뉴스 데이터 구조 분석 및 추출  $('.sa\_text').slice(0, 10).each((i, el) => {  const title = $(el).find('.sa\_text\_title').text().trim();  const content = $(el).find('.sa\_text\_lede').text().trim();  const publishedAt = $(el).find('.sa\_text\_press').text().trim();  let url = $(el).find('.sa\_text\_title a').attr('href') || '';    // AI-Agent가 URL 정규화 수행  if (url) {  if (url.startsWith('/')) {  url = 'https://news.naver.com' + url;  } else if (!url.startsWith('http')) {  url = 'https://news.naver.com' + url;}}    if (title && content && url) {  articles.push({  title,  content,  category: cat,  publishedAt,  url});}});    console.log(`AI-Agent 수집 완료 - ${cat}: ${articles.length}개 기사`);  allNews = allNews.concat(articles);    } catch (error) {  console.error(`AI-Agent ${cat} 크롤링 오류:`, error.message);}}    // AI-Agent가 중복 뉴스 제거 (제목 기준)  const uniqueNews = Array.from(new Map(allNews.map(item => [item.title, item])).values());  console.log(`AI-Agent 최종 수집: ${uniqueNews.length}개 고유 뉴스`);    // AI-Agent가 랜덤 섞기로 다양성 보장  return shuffleArray(uniqueNews);}  // AI-Agent가 수행하는 간단 요약 생성  function simpleSummary(content) {  if (!content) return '';    const sentences = content.split(/[.!?]/).filter(s => s.trim().length > 10);    if (sentences.length > 0) {  return sentences[0].trim() + '.'; }    if (content.length > 100) {  return content.substring(0, 100).trim() + '...'; }    return content;} |

### ****AI-Agent 개인화 시스템의 특징:****

* **실시간 가중치 해석**: EQ 슬라이더 변경을 즉시 개인화 알고리즘에 반영
* **지능형 카테고리 선별**: 0% 가중치 카테고리 자동 제외
* **다양성 보장**: 가중치 적용 후에도 균형잡힌 뉴스 제공
* **영구 저장**: 사용자 설정을 localStorage에 저장하여 재방문 시 복원
* **오류 복원력**: 설정 로드 실패 시 기본값으로 안전하게 복원

**AI-Agent 통합 시스템의 특징:**

* **다단계 AI 처리**: 뉴스 분석 → 스토리 변환 → 언어 번역 → 이미지 생성 복합 파이프라인
* **오류 처리 시스템**: API 실패 시 대체 콘텐츠 자동 제공
* **환경 적응성**: 개발/프로덕션 환경에 따른 동적 처리
* **모듈화 설계**: 각 AI 기능을 독립적으로 호출 가능한 구조

## 4.2 화면 구현 결과

### 4.2.1 메인 뉴스 피드 화면

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI가 생성한 콘텐츠는 부정확할 수 있습니다.**구현된 주요 UI 컴포넌트:**

* 상단 브랜딩 헤더 (nu:speak 로고 + 네비게이션)
* 카테고리 선택 탭 (추천, IT, 경제, 사회, 정치, 세계, 생활/문화)
* 개인화된 뉴스 카드 그리드 레이아웃
* 하단 EQ 슬라이더 고정 영역

**뉴스 카드 상세 구현:**

* 카테고리별 색상 구분
* AI 요약 미리보기
* 음성 듣기, 4컷 만화 보기 액션 버튼

### 4.2.2 EQ 슬라이더 설정 화면

**텍스트, 스크린샷, 소프트웨어, 디자인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**핵심 UX 구현 요소:**

* 카테고리별 슬라이더 실시간 조작
* 가중치 변경에 따른 뉴스 개수 변화 미리보기
* 설정 저장 기능
* 이전 화면으로 돌아가기 네비게이션

### 4.2.3 뉴스 상세 및 AI 콘텐츠 화면

**텍스트, 스크린샷, 웹 페이지, 웹사이트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**멀티모달 콘텐츠 구현:**

* 원문 뉴스와 3줄 요약 토글 표시
* 4컷 만화 이미지 및 각 컷별 설명
* TTS 음성 재생 컨트롤러

## 4.3 기술적 구현 성과

### 4.3.1 성공적으로 구현된 핵심 기능

**개인화 시스템**

* EQ 슬라이더를 통한 실시간 개인화
* 디바운싱 기법을 활용한 성능 최적화

**AI 콘텐츠 생성 파이프라인**

* OpenAI GPT-3.5를 활용한 뉴스 요약
* DALL-E 3 기반 4컷 만화 자동 생성
* Google TTS API를 통한 음성 변환

**사용자 인터페이스**

* React + TypeScript 기반 반응형 웹 인터페이스
* 직관적이고 일관된 사용자 경험

### 4.3.2 기술적 도전과 해결

**AI API 응답 시간 최적화**

* 문제: AI 콘텐츠 생성 시 대기 시간 발생
* 해결: 비동기 처리와 로딩 인디케이터를 통한 사용자 피드백

**실시간 개인화 구현**

* 문제: EQ 슬라이더 조작 시 과도한 API 호출
* 해결: 디바운싱 기법으로 API 호출 최적화

# 5. 향후 발전 가능성

## 5.1 미구현 기능 및 확장 아이디어

### 5.1.1 AI-Agent 대화형 시스템

**자연어 기반 뉴스 탐색**

* "오늘 IT 뉴스 알려줘"와 같은 자연어 질문 처리
* 사용자 의도 분석 및 맥락 이해 기반 응답
* 대화 히스토리를 활용한 연속적인 질의응답

**구현 방향성:**

* GPT-4를 활용한 고도화된 자연어 처리
* 사용자 개인화 정보와 대화 맥락 통합
* 실시간 뉴스 검색과 추천 시스템 연동

### 5.1.2 데이터베이스 연동 시스템

**사용자 행동 분석**

* 뉴스 클릭, 읽기 시간, 공유 패턴 데이터 수집
* 개인화 학습을 위한 사용자 히스토리 관리
* 추천 정확도 향상을 위한 피드백 시스템

**구현 방향성:**

* PostgreSQL 기반 사용자 데이터 관리
* Redis를 활용한 실시간 캐싱 시스템
* 머신러닝 기반 추천 알고리즘 고도화

### 5.1.3 플랫폼 확장 가능성

**모바일 애플리케이션**

* iOS/Android 네이티브 앱 개발
* 푸시 알림을 통한 개인화된 뉴스 제공

## 5.2 개발 과정에서 얻은 핵심 인사이트

### 5.2.1 AI-Agent 구현의 예상 외 복잡성

**멀티모달 통합의 어려움** 텍스트 요약, 이미지 생성, 음성 변환을 하나의 일관된 시스템으로 구현하는 것이 예상보다 훨씬 복잡했습니다. 각 모달리티 간의 정보 손실을 최소화하고 일관성을 유지하는 것이 가장 큰 도전이었습니다.

**프롬프트 엔지니어링의 중요성** AI-Agent가 "뉴스 분석 전문가", "창작 전문가" 등의 역할을 수행하도록 시스템 프롬프트를 설계하는 과정에서, AI의 응답 품질이 프롬프트 설계에 크게 좌우된다는 것을 실감했습니다.

### 5.2.2 사용자 중심 설계의 새로운 관점

**EQ 슬라이더의 직관성** 복잡한 개인화 시스템을 EQ라는 직관적 개념으로 단순화했을 때, 사용자가 시스템을 쉽게 이해하고 통제할 수 있다는 것을 확인했습니다. 이는 AI 시스템의 복잡함을 사용자에게 노출하지 않는 것의 중요성을 보여줍니다.

**실시간 피드백의 힘** 사용자가 EQ 슬라이더를 조작할 때마다 즉시 반영되는 시스템을 통해, 사용자-AI 간의 새로운 협력 패러다임을 만들 수 있었습니다.

### 5.2.3 팀 협업의 새로운 방식

**AI-Agent 중심 역할 분담** 전통적인 Frontend/Backend 구분이 아닌, AI-Agent의 역할과 능력을 중심으로 한 협업 방식이 더 효과적이었습니다. "어떤 기능을 만들까?"보다 "AI-Agent가 사용자를 어떻게 도울 수 있을까?"를 중심으로 한 접근이 혁신적인 결과를 만들어냈습니다.

## 5.3 프로젝트 완료 소감

nu:speak 프로젝트는 단순한 뉴스 서비스 개발을 넘어서, AI-Agent가 어떻게 사용자의 일상적인 정보 소비를 혁신할 수 있는지를 탐구하는 의미 있는 여정이었습니다.

특히 AI-Agent를 시스템의 중심에 두고 모든 기능을 설계했을 때, 기존의 기능 중심 접근법과는 완전히 다른 사용자 경험이 만들어진다는 것을 확인할 수 있었습니다. EQ 슬라이더를 통한 실시간 개인화, AI-Agent 기반 멀티모달 콘텐츠 생성, 그리고 사용자와 AI-Agent 간의 협력적 상호작용은 정보 과부하 시대의 새로운 해답을 제시했습니다.

이번 프로젝트를 통해 AI-Agent 시스템 설계부터 실제 구현까지의 전 과정을 경험할 수 있었으며, 향후 AI-Agent 중심의 서비스가 어떤 방향으로 발전해야 할지에 대한 명확한 비전을 얻을 수 있었습니다. nu:speak는 AI-Agent와 인간이 협력하여 더 나은 정보 경험을 만들어가는 미래 서비스의 출발점이 될 것입니다.

**부록**

**A. 기술 용어 정의**

| **용어** | **정의** |
| --- | --- |
| EQ (Emotional Quotient) | nu:speak에서 사용하는 카테고리별 관심도 조절 시스템. 사용자가 실시간으로 뉴스 카테고리별 가중치를 0-100% 범위에서 설정할 수 있는 개인화 기능 |
| 개인화 점수 | 사용자의 EQ 설정과 카테고리를 종합하여 각 뉴스에 부여되는 추천 적합도 점수 |
| 멀티모달 콘텐츠 | 텍스트, 이미지, 음성 등 다양한 형태로 제공되는 뉴스 콘텐츠 (요약, 4컷만화, TTS 등) |

**B. 시스템 요구사항**

**개발 환경:**

* Node.js v18.x 이상
* npm v9.x 이상
* Git v2.30 이상
* 필수 API 키: OpenAI API, Google TTS API

**실행 환경:**

* 모던 웹 브라우저 지원 (Chrome, Firefox, Safari, Edge)
* 모바일 기기 터치 인터페이스 지원

**C. API 키 설정 가이드**

**환경 변수 설정:**

bash

*# .env 파일 설정*

OPENAI\_API\_KEY=your\_openai\_api\_key

GOOGLE\_TTS\_API\_KEY=your\_google\_tts\_key

**보안 고려사항:**

* API 키는 환경 변수로만 관리
* 프로덕션 환경에서는 보안 저장소 사용 권장
* 정기적인 API 키 로테이션 실시

© 2025 nu:speak Development Team. All Rights Reserved.