SW 상세설계서 - 1팀 (팀백)

목차

---

1. 서론

1.1 목적 (Purpose)

이 설계서의 목적은 **Thesisfy** 시스템의 각 구성 요소를 구체적으로 설계하여 개발자들이 소프트웨어를 효율적이고 일관되게 구현할 수 있도록 돕는 것입니다. 이를 통해 시스템의 주요 기능, 데이터 흐름, 인터페이스, 보안 요구사항을 명확히 정의합니다.

1.2 범위 (Scope)

이 설계서는 **Thesisfy**의 백엔드 및 iOS 프론트엔드 모듈 설계를 포함하며, 시스템의 기능적 요구사항과 비기능적 요구사항을 모두 다룹니다. 또한, 데이터베이스 구조, 보안 설계, 성능 최적화, 테스트 계획 등을 포함하여 전반적인 시스템 개발을 위한 가이드를 제공합니다.

1.3 참고 문서 (Reference Documents)

- **Thesisfy** 프로젝트 계획서
- 소프트웨어 요구사항 명세서 (SRS)
- 시스템 아키텍처 설계 문서
- Node.js, MySQL 및 Swift 관련 기술 문서
- GDPR 및 관련 보안 지침

2. 시스템 개요

2.1 기능 설명 (Functional Overview)

Thesisfy는 사용자의 자연어 질문을 이해하고, 그 질문에 맞는 논문, 리포트, 기사 등을 정확하고 신속하게 찾아주는 AI 기반 학술 정보 검색 도구입니다. 주요 기능은 다음과 같습니다:

- **사용자 관리**:
  - 로그인 및 회원가입
  - 사용자 프로필 관리
- **학술 정보 제공**:
  - AI 기반 논문 검색 및 필터링
  - 논문 열람 및 자동 요약
- **AI 시스템**:
  - LLM을 활용한 논문 요약 및 핵심 내용 제공
- **보안 및 데이터 보호**:
  - JWT 인증을 통한 사용자 인증 및 권한 관리
  - GDPR 준수를 통한 개인정보 보호
- **알림 및 피드백**:
  - 푸시 알림을 통해 새로운 논문 업데이트 및 관심 논문 알림 제공

2.2 시스템 아키텍처 (System Architecture)

시스템은 클라이언트-서버 아키텍처로 구성되며, 다음과 같은 주요 컴포넌트로 나뉩니다:

- **사용자 인터페이스 (UI)**: iOS 애플리케이션의 사용자 인터페이스로, 사용자가 논문 검색 및 요약을 요청하고 결과를 확인할 수 있는 인터페이스를 제공합니다.
- **서버**: Node.js를 기반으로 사용자의 요청을 처리하고, AI 시스템 및 데이터베이스와의 통합을 통해 비즈니스 로직을 구현합니다.
- **데이터베이스**: MySQL을 사용하여 사용자 정보, 논문 데이터, 검색 히스토리, 추천 데이터 등을 관리하며 CRUD 작업을 수행합니다.
- **AI 서버**: 오케스트로에서 제공한 LLM API를 활용해 텍스트 분석 및 응답을 생성하여 사용자의 질문에 맞는 논문 요약 및 핵심 내용을 제공합니다.

3. 모듈 설계

3.1 사용자 관리 모듈 (User Management Module)

- **기능 설명**: 회원가입, 로그인, 프로필 관리 기능을 제공합니다.
- **입력 (Input)**: 사용자 정보 (이메일, 비밀번호, 이름 등)
- **출력 (Output)**: 인증 토큰, 프로필 정보
- **프로세스 흐름**:
  1. 사용자가 회원가입 정보를 제출하면, 서버는 이를 데이터베이스에 저장합니다.
  2. 비밀번호는 bcrypt 해시 알고리즘을 통해 암호화됩니다.
  3. 로그인 시, 사용자가 입력한 비밀번호와 해시된 비밀번호를 비교하여 인증합니다.
- **주요 클래스 및 메서드**:
  - **UserController**:
    - `registerUser(UserDto user)`: 회원가입 기능
    - `login(String email, String password)`: 로그인 기능
  - **UserService**:

        - `validateUserCredentials()`: 사용자 자격 검증
        - `createUser()`: 사용자 생성
    - **UserRepository**:
        - `save(UserEntity user)`: 사용자 정보 저장
        - `findByEmail(String email)`: 이메일로 사용자 검색

3.2 챗봇 논문 요약 및 검색 모듈 (Chatbot Summary and Search Module)

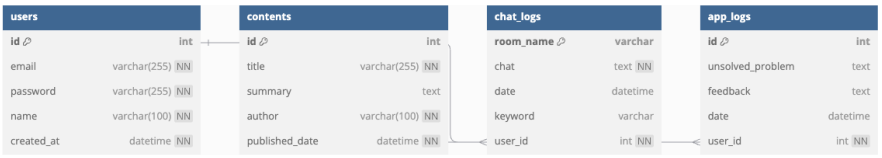
- **기능 설명**: 사용자의 자연어 질문을 이해하고, AI 챗봇이 해당 질문에 맞는 논문을 요약해 주고 관련 자료를 찾아줍니다.
- **입력 (Input)**: 사용자 질문 (자연어 형태)
- **출력 (Output)**: 논문 요약, 논문 목록 (제목, 저자, 요약, 논문 링크 등)
- **프로세스 흐름**:
    1. 사용자가 자연어로 질문을 입력합니다.
    2. AI 챗봇이 NLP(자연어 처리) 모델을 통해 질문의 의도를 파악합니다.
    3. 질문에 맞는 논문을 검색하고, 논문의 핵심 내용을 요약하여 사용자에게 보여줍니다.
    4. 추가로 관련 논문 목록을 제공하여 사용자가 원하는 자료를 선택할 수 있도록 합니다.
- **주요 클래스 및 메서드**:
    - **ChatbotController**
        - `processQuery(String query)`: 사용자 질문을 처리하여 결과 반환.
    - **ChatbotService**
        - `analyzeIntent(String query)`: 사용자의 질문을 분석하여 검색 의도 파악.
        - `fetchAndSummarizeContent(String query)`: 검색된 논문에 대해 요약 생성.
    - **ChatbotRepository**
        - `findRelevantPapers(String keywords)`: 키워드 기반으로 논문 검색.
        - `getSummaryById(int paperId)`: 논문 ID에 따라 요약된 정보 반환.

3.3 논문 검색 및 열람 모듈 (Paper Search and Access Module)

- **기능 설명**: 검색된 논문에 대한 전체 내용을 열람할 수 있는 URL을 제공하는 모듈입니다.
- **입력 (Input)**: 논문 ID 또는 논문 선택 (사용자가 선택한 논문)
- **출력 (Output)**: 전체 논문 열람 URL
- **프로세스 흐름**:
    1. 사용자가 논문 목록에서 원하는 논문을 선택합니다.
    2. 선택한 논문에 대한 전체 URL을 조회하여 제공합니다.
    3. 사용자는 제공된 링크를 통해 논문을 열람할 수 있습니다.
- **주요 클래스 및 메서드**:
    - **SearchController**
        - `viewPaperDetails(int paperId)`: 논문 상세 정보를 반환.
        - `getFullPaperUrl(int paperId)`: 논문의 전체 열람 URL 제공.
    - **SearchService**
        - `getPaperById(int paperId)`: 선택된 논문에 대한 상세 정보 조회.
    - **SearchRepository**
        - `fetchPaperDetails(int paperId)`: 논문 ID로 논문 데이터 조회.
        - `fetchPaperUrl(int paperId)`: 논문 ID로 전체 논문 URL 조회.

---

4. 데이터베이스 설계




```
-- 사용자 테이블 (User Table)
CREATE TABLE users (
    id INT AUTO_INCREMENT PRIMARY KEY COMMENT '사용자 고유 ID (자동 증가)',
    email VARCHAR(255) NOT NULL UNIQUE COMMENT '사용자 이메일 (유니크)',
    password VARCHAR(255) NOT NULL COMMENT '암호화된 비밀번호',
    name VARCHAR(100) NOT NULL COMMENT '사용자 이름',
    created_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '계정 생성 시간'
);

-- 논문 테이블 (Content Table)
CREATE TABLE contents (
    id INT AUTO_INCREMENT PRIMARY KEY COMMENT '논문 고유 ID (자동 증가)',
    title VARCHAR(255) NOT NULL COMMENT '논문 제목',
    summary TEXT COMMENT '논문 요약',
    author VARCHAR(100) NOT NULL COMMENT '논문 저자',
    published_date DATETIME NOT NULL COMMENT '논문 출판 날짜'
);

-- 채팅 로그 테이블 (Chat Log Table)
CREATE TABLE chat_logs (
    room_name VARCHAR(255) PRIMARY KEY COMMENT '채팅방 고유코드 (랜덤 생성)',
    chat TEXT NOT NULL COMMENT '대화 내용 작성',
    date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '채팅 시작한 날짜 저장',
    keyword VARCHAR(255) COMMENT 'LLM이 생성한 토픽이나 키워드 저장',
    user_id INT NOT NULL COMMENT '사용자 ID',
    FOREIGN KEY (user_id) REFERENCES users(id)
);

-- 앱 로그 테이블 (App Log Table)
CREATE TABLE app_logs (
    id INT AUTO_INCREMENT PRIMARY KEY COMMENT '로그 고유 ID (자동 증가)',
    unsolved_problem TEXT COMMENT 'LLM이 대답하지 못한 질문들 저장',
    feedback TEXT COMMENT '사용자 피드백',
    date DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP COMMENT '앱 출입 시간 기록',
    user_id INT NOT NULL COMMENT '사용자 ID',
    FOREIGN KEY (user_id) REFERENCES users(id)
);
```

4.1 사용자 테이블 (User Table)

컬럼명	데이터 타입	길이	기본값	PK	FK	not NULL	설
ID	INT	-	-	Y	N	A	사 (자
Email	VARCHAR	255	-	N	N	A	사 나
Password	VARCHAR	255	-	N	N	A	암 호
Name	VARCHAR	100	-	N	N	A	사

Created_at	DATETIME	-	CURRENT_TIMESTAMP	N	N	A	겨

4.2 논문 테이블 (Content Table)

컬럼명	데이터 타입	길이	기본값	PK	FK	not NULL	설명
ID	INT	-	-	Y	N	A	논문 고유 동 증가)
Title	VARCHAR	255	-	N	N	A	논문 제목
Summary	TEXT	-	-	N	N	D	논문 요약
Author	VARCHAR	100	-	N	N	A	논문 저자
Published_date	DATETIME	-	-	N	N	A	논문 출판

4.3 Chat Log 테이블 (Chat Log table)

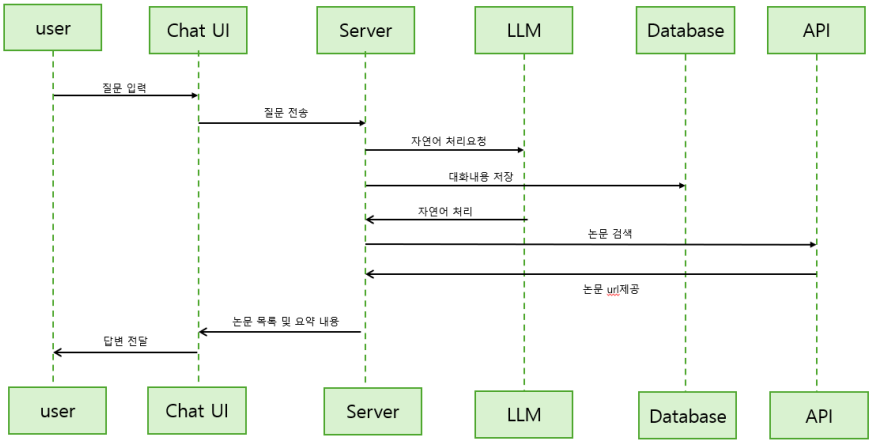
컬럼명	데이터 타입	길이	기본값	PK	FK	not NULL	설명
Room name	varchar	-	-	Y	N	A	채팅방 고 (랜덤 생성
Chat	varchar	-	-	N	N	D	대화 내용
Date	varchar	-	-	N	N	A	채팅 시작 저장
Keyword	varchar	-	-	N	N	D	LLM이 생 픽이나 키 장
Nickname	varchar	-	-	N	N	A	사용자 닉
(빈 컬럼)	varchar	n	-	N	D	-	(빈 컬럼에 설명 작성

4.4 App Log 테이블 (App Log table)

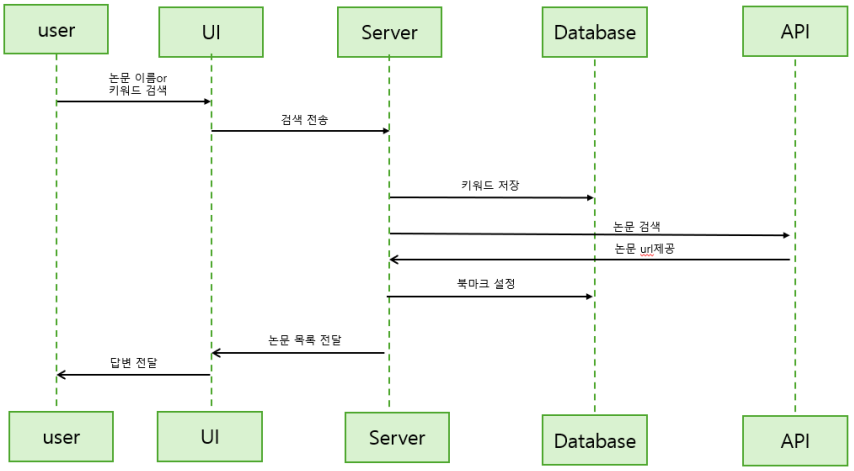
컬럼명	데이터 타입	길이	기본값	PK	FK	not NULL	설명
Unsolved_problem	varchar	-	-	N	N	A	LLM이 못한 질
Feedback	varchar	-	-	N	N	D	사용자
Date	varchar	-	-	N	N	A	앱 출입

5. 시퀀스 다이어그램

5.1 채팅형 논문 검색 시퀀스



5.2 논문 키워드 검색 시퀀스



6. API 설계

6.1 사용자 관리 API

- **회원가입 API**
  - **URL**: `/api/users/register`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "email": "user@example.com",
    "password": "password123",
    "name": "John Doe"
}
```

  - **Response**:

```
// json
{
    "message": "User registered successfully",
    "userId": 1
}
```

  - **에러 처리**:
    - 이메일 중복 시 400 에러 반환
    - 유효성 검사 실패 시 422 에러 반환
- **로그인 API**
  - **URL**: `/api/users/login`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "email": "user@example.com",
    "password": "password123"
}
```

- **Response**:

```
// json
{
    "message": "Login successful",
    "token": "jwt-token-here"
}
```

- **에러 처리**:
  - 이메일 또는 비밀번호 불일치 시 401 에러 반환

- **프로필 조회 API**
  - **URL**: `/api/users/profile`
  - **Method**: `GET`
  - **Headers**: `Authorization: Bearer {token}`
  - **Response**:

```
// json
{
    "userId": 1,
    "email": "user@example.com",
    "name": "John Doe",
    "nickname": "Scholar123",
    "job": "Graduate Student",
    "affiliation": "XYZ University"
}
```

6.2 학술 정보 제공 API

- **논문 검색 API**
  - **URL**: `/api/contents/search`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "query": "Impact of AI in education",
    "filters": {
        "type": "thesis",
        "date_range": {
            "from": "2020-01-01",
            "to": "2024-12-31"
        }
    }
```

```
}
```

- **Response**:

```
// json
{
    "message": "Search completed",
    "results": [
        {
            "id": 101,
            "title": "AI's Role in Modern Education",
            "summary": "This paper explores...",
            "author": "Jane Doe",
            "published_date": "2023-06-15",
            "source_url": "https://example.com/thesis123"
        },
        ...
    ]
}
```

- **논문 열람 및 요약 API**
  - **URL**: `/api/contents/{id}`
  - **Method**: `GET`
  - **Response**:

```
// json
{
    "id": 101,
    "title": "AI's Role in Modern Education",
    "full_text": "...",
    "summary": "This paper explores..."
}
```

6.3 AI 시스템 API

- **논문 요약 API**
  - **URL**: `/api/ai/summary`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "contentId": 101
}
```

  - **Response**:

```
// json
{
    "summary": "This paper explores..."
```

```
}
```

6.4 보안 및 데이터 보호 API

- **JWT 인증 API**
  - **URL**: `/api/auth/verify`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "token": "jwt-token-here"
}
```

  - **Response**:

```
// json
{
    "message": "Token is valid"
}
```

6.5 알림 및 피드백 API

- **논문 알림 API**
  - **URL**: `/api/notifications/subscribe`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "userId": 1,
    "contentId": 101
}
```

  - **Response**:

```
// json
{
    "message": "Subscribed to content notifications successfully"
}
```

- **피드백 제출 API**
  - **URL**: `/api/feedback`
  - **Method**: `POST`
  - **Request Body**:

```
// json
{
    "userId": 1,
```

```
    "contentId": 101,
    "feedback": "This article was very insightful!"
}
```

- **Response**:

```
// json
{
    "message": "Feedback submitted successfully"
}
```

7. 보안 설계

보안 항목	설명	보안 대책	구현 단계
1. 사용자 인증 및 권한 관리	JWT(JSON Web Token)를 사용하여 안전하게 사용자 인증과 세션 관리를 수행합니다. 사용자 역할(관리자, 일반 사용자)에 따라 권한을 구분하여 관리합니다.	- JWT를 사용하여 로그인 시 토큰을 발급하고 API 요청 시 인증 및 권한을 검증- 역할 기반 접근 제어(Role-Based Access Control, RBAC)를 통해 사용자의 권한에 따라 기능 접근을 제한	1. 로그인/회원가입 API에 JWT 발급 로직 추가2. API 호출 시 JWT 인증 필터 추가3. RBAC 로직으로 사용자 역할 구분
2. 비밀번호 암호화	사용자의 비밀번호는 안전하게 암호화하여 데이터베이스에 저장하고 인증 시 비교합니다.	- 비밀번호는 bcrypt 알고리즘을 사용하여 해시 처리 후 저장- 인증 시 사용자가 입력한 비밀번호와 해시된 비밀번호를 비교하여 검증	1. bcrypt 라이브러리를 사용하여 비밀번호 해시 처리2. 로그인 API에서 비밀번호 비교 로직 구현
3. 데이터 전송 보안	HTTPS를 사용하여 클라이언트와 서버 간의 모든 통신을 암호화합니다.	- SSL/TLS 인증서를 적용하여 모든 API와 클라이언트-서버 간 통신을 HTTPS로 보호- 중요한 정보가 전송될 때는 추가적인 암호화 알고리즘(AES)을 사용하여 이중 보호	1. 서버에 SSL 인증서 적용 및 HTTPS 설정2. 데이터 전송 시 AES 암호화 모듈 적용
4. 입력값 검증 및 보호	SQL 인젝션, XSS 등 공격을 방지하기 위해 모든 입력값을 검증하고 필터링합니다.	- 모든 사용자 입력값에 대해 서버 및 클라이언트 측 검증 로직 구현- SQL 인젝션 방지를 위해 파라미터화된 쿼리 사용 및 ORM 활용- HTML, JS 응답 시 XSS 필터링 및 인코딩 처리	1. 클라이언트 측에서 기본적인 유효성 검사 구현2. 서버에서 파라미터화된 쿼리와 ORM을 통한 SQL 인젝션 방어3. XSS 방지 필터 적용
5. 데이터 저장 보안	사용자 정보, 검색 기록 등 민감한 데이터는 안전하게 저장됩니다.	- 사용자의 민감한 정보는 데이터베이스에 AES 암호화를 사용하여 저장- 검색 기록은 암호화된 상태로 관리하여 사용자 프라이버시 보호	1. AES 암호화 알고리즘을 적용하여 민감한 데이터 암호화 저장2. 검색 기록 저장 로직에 암호화 적용
6. 파일 업로드 보안	사용자 프로필 이미지와 같은 파일 업로드 기능에서 악성 파일 업로드를 방지합니다.	- 업로드된 파일의 확장자 및 파일 크기 제한- 실행 가능한 파일 업로드 차단- 업로드된 파일은 임의의 파일명으로 저장하고, 외부에서 접근할 수 없는 경로에 저장	1. 파일 업로드 시 확장자와 크기 제한 로직 구현2. 저장 경로 설정 및 랜덤 파일명 생성 로직 구현
7. 웹 서비스 요청 및 결과 검증	API 요청 및 응답에서 사용자 데이터를 안전하게 처리합니다.	- CSRF(Cross-Site Request Forgery) 방지를 위해 CSRF 토큰 사용- 모든 API 요청에 대해 요청 검증 로직을 포함하고, 잘못된 요청에 대해서는 오류 메시지를 반환	1. API 요청 시 CSRF 토큰 검증 로직 추가2. 잘못된 요청에 대해 400/403 에러 코드 반환
8. 중요 정보 전송 및 보호	중요 정보를 네트워크로 전송할 때 암호화하여 안전하게 보호합니다.	- 세션 관리에 사용되는 토큰은 반드시 HTTPS를 통해 전송- 중요한 데이터(예: 비밀번호 변경 요청)는 AES 암호화 후 전송	1. HTTPS 적용 후 토큰 전송 구현2. 비밀번호 변경 및 민감한 데이터 전송 시 추가적인 AES 암호화 적용
9. 권한 검증 및 중요 기능 보호	비밀번호 변경, 결제 등 중요 기능은 적절한 권한 검증을 통해 보호됩니다.	- 중요 기능 호출 시마다 사용자 권한 및 유효성을 검증- CSRF 방지를 위해 중요 기능 수행 시 추가 검증 토큰 요구	1. 중요 기능 API에 권한 검증 로직 포함2. CSRF 토큰 기반 추가 검증 로직 적용
10. 모니터링 및 로그 관리	시스템 내 발생하는 보안 이벤트와 오류를 기록하고 모니터링합니다.	- 인증 실패, 비정상적인 API 접근, 데이터 변조 시도 등의 보안 이벤트를 로그로 기록- 실시간 모니터링 시스템(New Relic 등)을 통해 서버 상태를 감시	1. 시스템 로그 파일 저장 및 보안 이벤트 로깅 구현2. 모니터링 시스템 통합 및 실시간 알림 설정

8. 성능 및 확장성 고려사항

8.1 성능 고려사항

- **빠른 응답 속도**: 사용자 요청에 대한 신속한 응답을 위해 챗봇의 검색 알고리즘을 최적화하고, 캐싱 기술을 사용하여 자주 검색되는 논문 데이터를 빠르게 제공해야 합니다.
- **데이터베이스 최적화**: 대량의 논문 데이터를 저장하고 검색할 수 있도록 데이터베이스 설계를 효율적으로 하고, 인덱싱을 통해 검색 속도를 향상시킵니다. 관계형 데이터베이스의 경우, 적절한 인덱스 및 쿼리 최적화가 필수입니다.
- **부하 분산**: 트래픽이 많아질 경우를 대비해, 로드 밸런싱을 통해 여러 서버로 요청을 분산시켜 성능을 유지합니다. AWS, GCP 등 클라우드 서비스의 로드 밸런싱 서비스를 활용할 수 있습니다.
- **캐싱**: Redis나 Memcached와 같은 인메모리 캐싱 시스템을 사용해 반복적으로 요청되는 데이터에 대해 빠른 접근을 제공합니다.
- **API 속도 최적화**: 외부 API와의 통신 속도를 최적화하기 위해 비동기 처리나 적절한 요청 큐를 사용하여 API의 대기 시간을 줄입니다.

8.2 확장성 고려사항

- **모듈화된 아키텍처**: 챗봇의 각 기능을 독립적인 모듈로 설계하여 필요에 따라 기능을 추가하거나 교체할 수 있게 합니다. 예를 들어, 검색 알고리즘, 데이터 수집 모듈, 사용자 인터페이스를 분리하여 관리할 수 있도록 합니다.
- **수평적 확장성**: 서버나 데이터베이스를 쉽게 확장할 수 있도록 오케스트로 클라우드 기반 인프라를 이용할 예정입니다.
- **마이크로서비스 아키텍처**: 챗봇의 각 기능을 독립적인 마이크로서비스로 개발하여 서로 독립적으로 확장할 수 있게 합니다. 예를 들어, 사용자 관리, 검색 기능, 데이터 수집 기능 등을 각기 다른 서비스로 분리할 수 있습니다.
- **데이터베이스 샤딩**: 대용량 데이터를 처리할 수 있도록 샤딩을 사용하여 데이터베이스를 여러 서버에 분산합니다. 특히 데이터 양이 방대할 경우, 논문 ID 등을 기준으로 데이터를 나누어 각 서버에서 처리하도록 설계합니다.

8.3 추가 고려사항

- **API 요청 제한 관리**: 외부 API를 사용하는 경우, API 요청 제한(쿼터)을 고려하여 트래픽 제어를 위해 요청 수를 관리할 수 있는 로직을 구현합니다.
- **사용자 데이터 및 보안 관리**: 사용자 데이터를 안전하게 처리하고 저장하기 위해 SSL 암호화를 통해 보안을 강화하고, 필요에 따라 개인 정보 보호 규정에 따라 데이터를 관리합니다.
- **모니터링 및 알림 시스템**: 시스템 성능과 오류를 실시간으로 모니터링하고, 장애가 발생했을 때 알림을 받을 수 있도록 시스템 상태를 체크할 수 있는 모니터링 툴을 설치합니다. 예를 들어, Prometheus와 Grafana를 활용한 모니터링을 구현할 수 있습니다.

9. 테스트 계획

- **유닛 테스트**:
  - 주요 서비스 클래스의 개별 기능에 대한 유닛 테스트를 수행합니다.
  - 예: 사용자 생성 기능의 유효성 검사, 논문 검색 알고리즘의 정확성 테스트.
- **통합 테스트**:
  - 모듈 간 연동 및 API 통합 테스트를 진행하여 시스템의 연계성을 검증합니다.
  - 예: 사용자 로그인 후 논문 검색 및 요약 기능이 올바르게 작동하는지 확인.
- **테스트 도구**:
  - JUnit, Mockito를 사용하여 백엔드 유닛 및 통합 테스트 수행.
  - Postman을 통해 API 호출 및 통합 테스트 진행.

10. 배포 계획

- **배포 환경**:
  - AWS EC2 인스턴스를 사용하여 서버와 데이터베이스를 배포합니다. Node.js 백엔드 서버와 MySQL 데이터베이스가 포함된 환경을 설정하여 안정적인 학술 정보 검색 시스템을 운영합니다.
- **배포 자동화**:

- **CI/CD 파이프라인**을 구축하여 GitHub Actions 또는 Jenkins를 사용해 자동 배포를 설정합니다.
- 코드 변경 시, 자동으로 빌드, 테스트합니다.
- **Docker**를 사용하여 애플리케이션을 컨테이너화하고, **AWS ECS**와 같은 컨테이너 오케스트레이션 서비스를 통해 확장성을 보장합니다.