[Chat Gpt를 이용한 그림책 제작 모바일**APP]**

요구사항 명세서

2023년 12월 09일

문서번호 : 23-Team1-11-Doc-002

소 속 : 충북대학교 소프트웨어학과

팀 명 : Team1-11

팀 원 : 예경민, 손지연, 조성민

지도교수 : 강재구, 이종연 교수님

제/개정 이력

버전	날짜	작성자 성명	제/개정사항	비 고
v1.0	23. 10. 06	예경민, 조성민, 손지연	8번 항목까지 작성	
v1.1	23. 10. 18	예경민, 조성민, 손지연	요구사항 변경 및 전체 수정	
v1.2	23.11.24	예경민, 조성민, 손지연	UML다이어그램 및 알고리즘 작성	
v1.3	23.12.04	예경민, 조성민, 손지연	1차 전체 문서 수정	
v1.4	23.12.09	예경민, 조성민, 손지연	2차 전체 문서 수정	
v.1.5	23.12.15	예경민, 조성민, 손지연	최종 전체 문서 수정	

목 차

1. 시스템 개요 **7**
1.1 개발목표 7
1.2 개발범위 및 내용 7
1.3 제품 가치 8
1.4 대상 고객 8
1.5 사용 시나리오 9
1.6 업무분석표(WBS) 10
1.7 gantt chart 11
1.8 용어 정의 11

2. 요구사항 분석 **12**
2.1 기능적 요구사항 12
2.1.1 사용사례 분석 12
2.1.2 데이터 처리 흐름도 12
2.1.3 시스템 작업내용 13
2.1.4 관리 기능 13
2.1.5 트랜잭션 처리 작업 13
2.1.6 모든 화면에 대해 수행되는 작업의 세부 사항 14
2.1.6.1 스토리 등록 기능 14
2.1.6.2 스토리 생성 기능 15
2.1.6.3 맞춤법 교정 기능 15
2.1.6.4 스토리 나누기 기능 15
2.1.6.5 스토리 요약 기능 15
2.1.6.6 그림 생성 기능 16
2.1.6.7 레이아웃 및 디자인 편집 기능 16
2.1.6.8 저장 및 공유 기능 16

2.2 비기능적 요구사항 17
2.2.1 성능 요구사항 17
2.2.2 신뢰성 요구사항 17
2.2.3 부하 및 스트레스 요구사항 17
2.2.4 보안 요구사항 17
2.2.5 운영(용량) 요구사항 17
2.2.6 사용성 요구사항 18
2.2.7 호환성 요구사항 18
2.2.8 안정성 및 가용성 요구사항 18
2.2.9 확장성 요구사항 18

2.2.10 유지보수성 요구사항 19
2.2.11 설치 요구사항 19
2.2.12 복구 요구사항 19
2.2.13 법적 규정 및 환경 요구사항 19
2.2.14 위험분석 및 제약 요구사항 19
2.2.15 문서 산출물 요구사항 19

3. 외부 인터페이스 요구사항 **20**
3.1 하드웨어 인터페이스 요구사항 20
3.2 소프트웨어 인터페이스 요구사항 20
3.3 통신 인터페이스 요구사항 20

4. 사용자 인터페이스 요구사항 설계 **21**
4.1 사용자 인터페이스 요구사항 21
4.2 사용자 인터페이스 설계(관리자용, 고객용, App용 등) 21
4.2.1 관리자용 사용자 인터페이스 설계 21
4.2.2 사용자용 사용자 인터페이스 설계 22

5. 기능적 모델링 **24**
5.1 사용사례 다이어그램 24
5.1.1 액터리스트 25
5.2 사용사례 명세(사용 시나리오) 26
5.2.1 Use Case: 스토리 작성 26
5.2.2 Use Case: 스토리 문단 선택 27
5.2.3 Use Case: 스토리 나누기 28
5.2.4 Use Case: 그림 생성 29
5.2.5 Use Case: 그림책 편집 30
5.2.6 Use Case: 그림책 관리 32
5.2.7 Use Case: 로그인 33
5.2.8 Use Case: 회원 정보 수정 34
5.2.9 Use Case: 회원 탈퇴 36
5.2.10 Use Case: 회원 가입 37

6. 구조적 모델링 **39**
6.1 클래스 다이어그램 39
6.2 CRC cards과 클래스 명세서 39
6.2.1 클래스 명세서 39
6.2.2 CRC cards 41
6.2.2.1 Class:User 41
6.2.2.2 Class:Book 42
6.2.2.3 Class:Book List 43

6.2.2.4 Class:스토리 44
6.2.2.5 Class:ChatGPT 45
6.2.2.6 Class:그림 45
6.2.2.7 Class:문단 46
6.3 Abstract Class와 interface명세 47
6.4 그 밖의 자료구조 명세 47

7. 데이터베이스 스키마 명세 **47**
7.1 User 스키마 테이블 명세서 47
7.2 Book 스키마 테이블 명세서 47
7.3 스토리 스키마 테이블 명세서 47
7.4 그림 스키마 테이블 명세서 48

8. 행위 모델링**:** 사용사례에 대한 시퀀스 다이어그램 **49**
8.1 '사용자 관리(사용자 조회, 검색, 삭제)에 대한 시퀀스 다이어그램 49
8.2 '회원관리(회원가입, 탈퇴, 회원 정보 수정)'에 대한 시퀀스 다이어그램 .. 50
8.3 '회원 정보 수정'에 대한 시퀀스 다이어그램 51
8.4 '서재조회'에 대한 시퀀스 다이어그램 52
8.5 '책 읽기'에 대한 시퀀스 다이어그램 53
8.6 '책 삭제'에 대한 시퀀스 다이어그램 53
8.7 '그림책 만들기'에 대한 시퀀스 다이어그램 54
8.8 '그림책 이어서 생성'에 대한 시퀀스 다이어그램1 55
8.8.1 '그림책 이어서 생성'에 대한 시퀀스 다이어그램2 56
8.9 '스토리 작성'에 대한 시퀀스 다이어그램 57
8.10 '스토리 나누기'에 대한 시퀀스 다이어그램 57
8.11 '문장 형태 변환'에 대한 시퀀스 다이어그램 58
8.12 '그림 생성'에 대한 시퀀스 다이어그램 58
8.13 '그림책 편집'에 대한 시퀀스 다이어그램 59

9. 클래스 메소드 알고리즘 명세 **60**
9.1 회원가입() 알고리즘 명세 60
9.2 회원정보저장() 알고리즘 명세 63
9.3 사용자검색() 알고리즘 명세 66
9.4 스토리 생성 및 맞춤법 검사() 알고리즘 명세 69
9.5 스토리수정() 알고리즘 명세 72
9.6 문장선택() 알고리즘 명세 75
9.7 회원정보수정() 알고리즘 명세 77
9.8 회원삭제() 알고리즘 명세 78
9.9 회원탈퇴() 알고리즘 명세 79

9.10 임시저장 여부조회() 알고리즘 명세 80
9.11 임시저장 위치 확인() 알고리즘 명세 80
9.12 책갈피 생성() 알고리즘 명세 81
9.13 책 읽기() 알고리즘 명세 82
9.14 이어서 읽기() 알고리즘 명세 83
9.15 처음부터 읽기() 알고리즘 명세 84
9.16 책 삭제() 알고리즘 명세 86
9.17 책 조회() 알고리즘 명세 87
9.18 책 저장() 알고리즘 명세 88
9.19 문단 나누기() 알고리즘 명세 89
9.20 그림_생성() 알고리즘 명세 90
9.21 책_글꼴_설정() 알고리즘 명세 92
9.22 텍스트_정렬() 알고리즘 명세 92
9.23 가운데_정렬() 알고리즘 명세 95
9.24 왼쪽_정렬() 알고리즘 명세 96
9.25 오른쪽_정렬() 알고리즘 명세 97
9.26 쪽번호_매기기() 알고리즘 명세 98
9.27 그림_배치() 알고리즘 명세 99
9.28 문단_배치() 알고리즘 명세 100
9.29 그림책_편집_요청() 알고리즘 명세 101
9.30 스토리_저장() 알고리즘 명세 103
9.31 문단_저장() 알고리즘 명세 104

10. 요구사항 검토 결과 **106**

11. 결론 **106**
11.1 부록 106
11.2 참고문헌 106

1. 시스템 개요

1.1 개발목표

Chat GPT를 활용하여 다양한 연령대가 쉽게 그림 창작물을 만들 수 있는 안드로이드 애플리케이션을 구현한다.

1.2 개발범위 및 내용

1) 스토리 생성

창작자는 그림책의 스토리 생성을 위해 아래의 내용들을 결정해야 한다.

- 어떤 주제로 그림책을 작성할 것인가
- 그림책에 어떤 캐릭터가 출연할 것인가
- 어떤 내용을 그림책에 담을 것인가

2) 맞춤법 교정

NPL api를 이용해 완성된 스토리의 맞춤법 교정을 진행한다.

3) 스토리 나누기

스토리가 완성되었다면 chat GPT를 이용해서 스토리에 적합한 그림을 생성해야한다. 그림 생성을 위해 스토리는 적당한 길이가 되어야 한다. 따라서, chat GPT를 이용해 스토리를 나누도록 한다. (창작자가 8개로 나눠줘 하면 8개로 나눠줘야됨)

4) 문장 선택

chat GPT 이용해 나뉘어진 스토리에서 각 문단에서 한 문장을 선택한다.

5) 그림 생성

Chat GPT의 그림 답변을 이용한다. 그림 답변은 프롬프트를 사용하여 진행된다.

그림은 다음과 같이 생성된다.

- NPL을 api 형태로 받아와 요약된 문장의 핵심적인 단어를 추출한다.
- 추출된 핵심적인 단어들은 프롬프트의 형태의 맞게 작성된다.

- 작성된 프롬프트를 통해 chat GPT는 그림 답변의 형태로 적합한 그림을 생성한다.

6) 그림과 스토리 결합하기
창작자는 생성된 그림중에 원하는 그림을 '선택'한다. 선택된 그림은 그림 생성 시에 사용된 스토리와 결합된다.

1.3 제품 가치

- 기능성: 창작자는 ID와 비밀번호를 사용하여 안전하게 로그인 할 수 있다.
- 유용성: 메뉴 및 버튼은 명확하게 레이블되어있으며, 창작자는 쉽게 원하는 정보를 찾을 수 있다.
- 신뢰성: 주기적인 데이터베이스 백업 작업이 이루어지며, 장애 발생 시 빠른 복구가 가능하다.

1.4 대상 고객

- 어린이: 그림책은 어린이들의 교육과 엔터테인먼트에 기여하며, 이런 앱은 그들의 창의력과 독해 능력을 향상시키는데 도움을 줄 수 있다.

- 부모 및 양육자: 부모 및 양육자들은 자녀의 독서 습관을 촉진하고, 교육적인 콘텐츠에 관심을 가지며 해당 앱은 부모들이 자녀들과 함께 독서를 즐기는데 도움을 줄 수 있다.

- 교사 및 교육 기관: 학교, 유치원, 학습 센터 등의 교육 기관에서 그림책 앱은 교육 자료로 활용될 수 있으며, 교사들은 학생들에게 창의적인 스토리텔링을 장려하는데 활용할 수 있다.

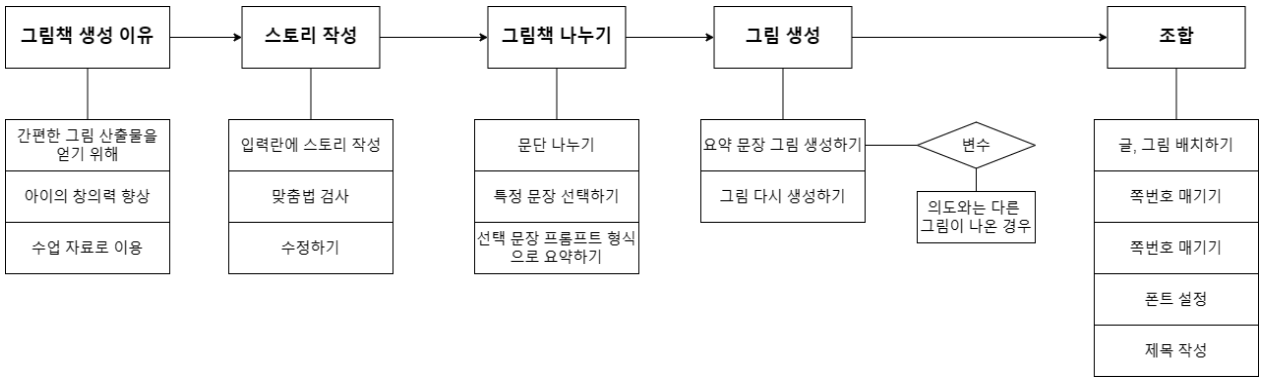
- 글쓰기 및 창작 활동에 관심 있는 사람들: 어린이와 양육자 외에도 창의적인 글쓰기나 이야기 만들기에 관심 있는 어른들이 이용할 수 있다.

1.5 사용 시나리오

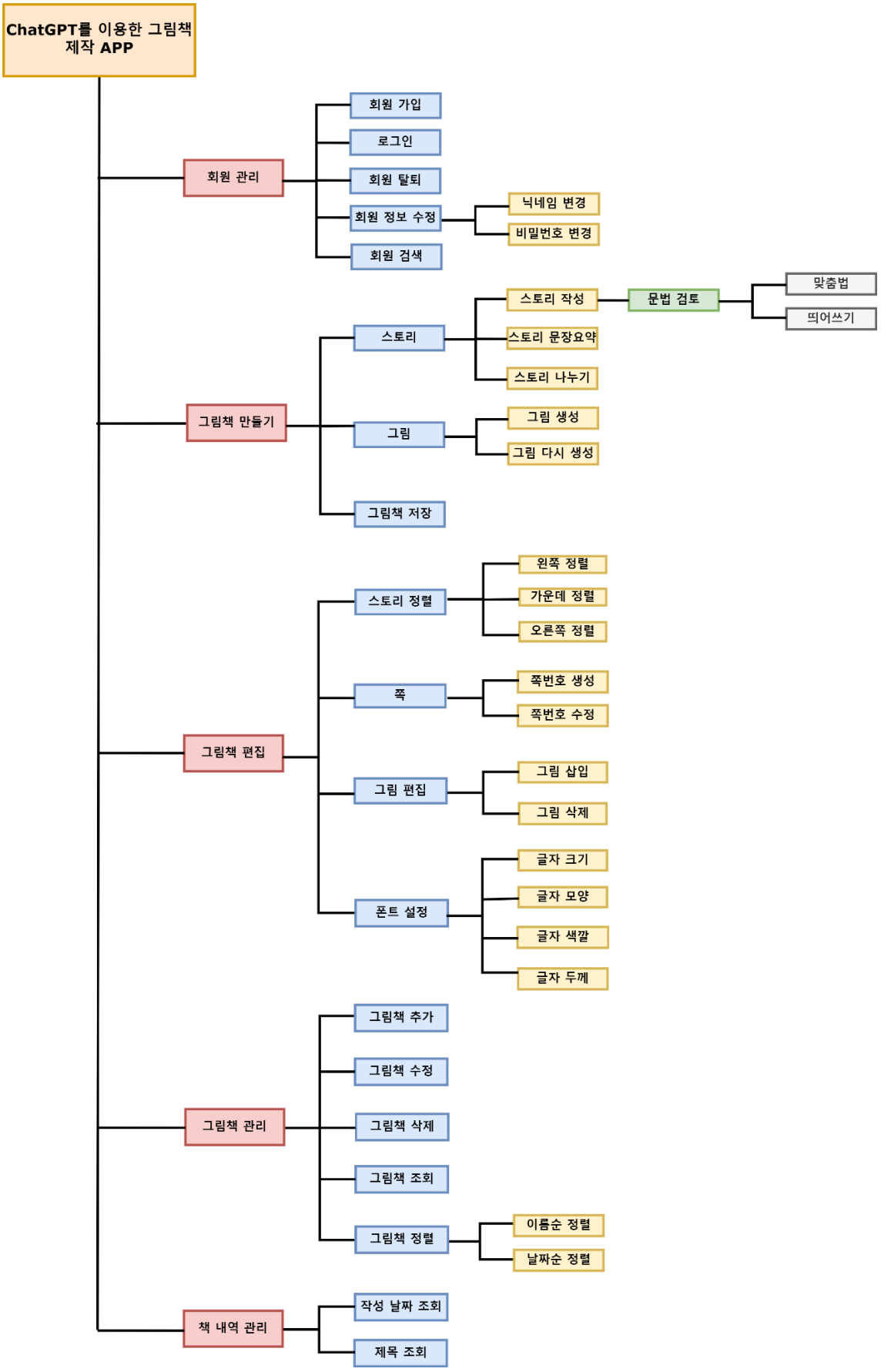
그림책 만들기 App은 개인이 작성한 글을 AI가 그에 맞는 그림을 생성하여 책으로 만들어 주는 서비스를 제공합니다.

이 애플리케이션은 간단하게 스토리 작성하기, 나누기, 그림 생성, 조합하는 4단계로 작성될 수 있습니다.

- 그림책 생성 이유 (비교적 간편하고 경제적인 그림산출물을 얻기 위해, 아이의 창의력 향상을 위해, 수업자료에 이용하기 위해)
- 그림책 나누기 (문단 나누기, 그림 생성할 문장 선택하기)
- 그림 생성하기 (chat GPT를 통해 그림 생성하기, 그림 다시 생성하기)
- 조합하기 (그림과 글을 페이지에 배치하기, 쪽번호 매기기, 정렬하기, 폰트 설정하기)



1.6 업무분석표(WBS)


ChatGPT를 이용한 그림책 제작 APP
회원 관리
회원 가입
로그인
회원 탈퇴
회원 정보 수정
닉네임 변경
비밀번호 변경
회원 검색
그림책 만들기
스토리
스토리 작성
문법 검토
맞춤법
띄어쓰기
스토리 문장요약
스토리 나누기
그림
그림 생성
그림 다시 생성
그림책 저장
그림책 편집
스토리 정렬
왼쪽 정렬
가운데 정렬
오른쪽 정렬
쪽
쪽번호 생성
쪽번호 수정
그림 편집
그림 삽입
그림 삭제
폰트 설정
글자 크기
글자 모양
글자 색깔
글자 두께
그림책 관리
그림책 추가
그림책 수정
그림책 삭제
그림책 조회
그림책 정렬
이름순 정렬
날짜순 정렬
책 내역 관리
작성 날짜 조회
제목 조회

1.7 Gantt Chart

프로젝트 추진 일정														
2023. 09 ~ 2024. 10														
단계		9월	10월	11월	12월	1월	2월	3월	4월	5월	6월	7월	8월	9월	10월
계획서작성															
자료조사/분석															
메인 메뉴 설계	History 설계														
	내 서재 설계														
	마이페이지 설계														
동화책 만들기 설계	스토리 생성 설계														
	스토리 나누기 설계														
	문장 형태 변환 설계														
	그림 생성 설계														
	그림책 편집 설계														
관리자 마이페이지 설계															
인공지능 학습															
메인 메뉴 구현	History 구현														
	내 서재 구현														
	마이페이지 구현														
동화책 만들기 구현	스토리 생성 구현														
	스토리 나누기 구현														
	문장 형태 변환 구현														
	그림 생성 구현														
	그림책 편집 구현														

1.8 용어 정의

본 문서의 이해를 돕기 위해 사용된 모든 용어 및 약어를 설명하고 정의합니다.

용어	설명
창작자	작품을 새롭게 만들어 내는 사람.
ChatGPT	ChatGPT란 OpenAI가 개발한 GPT-3.5 기반의 대형 언어 모델(large language model, LLM) 챗봇.
Stable Diffusion	Text를 Image로 변환해주는 AI
프롬프트	시스템이 다음 명령이나 메시지, 또는 다른 사용자의 행동을 받아들일 준비가 되었음을 사용자에게 알려주는 메시지.

2. 요구사항 분석

2.1 기능적 요구사항

2.1.1 사용사례 분석

- **<Usecase** 분석**>**
  - Actor : 사용자, 관리자, 시스템, Chat GPT
  - 요구사항: 회원 관리 기능, 그림책 생성 기능, 서재 조회 및 책 읽기 기능, 환경 설정 기능(마이페이지 설정)

- **<Usecase** 식별**>**
  - 회원 관리 기능:
    - 관리자가 등록된 회원의 정보를 조회, 삭제, 수정한다.
  - 마이페이지 설정 기능:
    - 사용자가 개인 정보 및 어플 환경을 설정한다.
  - 그림책 생성 기능:
    - 사용자는 그림책을 생성하기 위해 그림책의 스토리 작성, 문단 나누기, 문장 선택, 요약, 그림 생성, 편집한다.
    - 시스템은 사용자와 Chat GPT 사이에서 정보를 주고 받는다.
    - Chat GPT는 문단 나누기, 문장 요약, 그림을 생성한다.
  - 서재 조회:
    - 사용자는 서재에서 책 목록을 조회한다.
    - 사용자는 책 목록에서 읽을 책을 선택하여 책을 읽는다.
    - 시스템은 책 목록을 불러온다.

2.1.2 데이터 처리 흐름도

- 사용자는 그림책 작성을 시작하고 주제, 캐릭터, 내용을 입력한다.

- 입력된 내용은 NLP API를 통해 맞춤법 교정을 거친다.

- 사용자가 정한 기준에 따라 스토리를 Chat GPT로 나누고, 각 문단을 Chat GPT를 이용하여 요약한다.

- NLP API로부터 추출된 핵심 단어를 이용하여 Chat GPT가 그림을 생성한다.

- 사용자가 선택한 그림은 해당 그림을 생성할 때 사용된 스토리와 결합된다.

2.1.3 시스템 작업내용

- 그림책 작성: 사용자 입력을 받아 그림책의 주제, 캐릭터, 내용을 구성한다.

- 맞춤법 교정: NLP API를 이용하여 그림책의 맞춤법을 교정한다.

- 스토리 나누기: 사용자가 정한 기준에 따라 스토리를 Chat GPT로 나눈다.

- 문장 요약: Chat GPT를 이용하여 나뉜 스토리의 각 문단을 요약한다.

- 그림 생성: NLP API로부터 추출된 핵심 단어를 이용하여 Chat GPT가 그림을 생성한다.

- 그림과 스토리 결합: 사용자가 선택한 그림은 해당 그림을 생성할 때 사용된 스토리와 결합된다.

2.1.4 관리 기능

- 데이터베이스 관리: 시스템은 안전하게 사용자가 작성한 그림책 및 사용자 정보를 관리하는 데 필요한 데이터베이스를 구축하고 유지해야 하기에 이 데이터베이스는 무결성을 유지하고, 적절한 보안 기술을 적용하여 사용자의 개인 정보를 보호해야 합니다. 또한, 데이터베이스에 대한 백업 및 복구 전략이 필요하며, 시스템 장애 시에도 안정적으로 작동할 수 있어야 한다.

2.1.5 트랜잭션 처리 작업

- 그림 생성 트랜잭션: 그림 생성은 NLP API와 Chat GPT를 통한 다단계 프로세스로 이루어지며, 이 과정에서의 오류나 중단은 트랜잭션 처리를 통해 안정적으로 관리되어야 한다.

- 원자성 (Atomicity): 그림 생성 작업은 여러 단계로 이루어지지만, 이 과정에서 어떠한 단계에서든 문제가 발생하면 전체 작업을 롤백하여 시스템을 일관된 상태로 유지되어야 한다.

- 일관성 (Consistency): 트랜잭션이 완료된 후에 시스템은 언제나 일관된 상태여야 한다. 그림 생성이 정상적으로 이루어지면, 그 결과물과 스토리는 상호 일관성을 유지해야 한다.

- 격리성 (Isolation): 여러 사용자가 동시에 그림 생성을 시도할 경우에도 서로의 작업에 영향을 주지 않도록 격리시키며, 동시성 문제를 방지하고 안정성을 확보한다.

- 지속성 (Durability): 그림 생성이 성공적으로 완료되면 해당 결과는 시스템이 장애 상태에 놓여도 계속해서 유지되어야 한다. 즉, 데이터베이스에 영속적으로 저장되어야 하며, 시스템 복구 시에도 손실 없이 복구되어야 한다.

2.1.6 모든 화면에 대해 수행되는 작업의 세부 사항

2.1.6.1 스토리 등록 기능

2.1.6.1.1 사용자는 회원가입을 통해 계정을 생성할 수 있다.

2.1.6.1.2 사용자는 회원가입을 진행할 때, 아이디, 비밀번호, 닉네임, 이메일을 입력할 수 있다.

2.1.6.1.2.3 사용자는 로그인을 완료해야만 다른 기능들을 사용할 수 있다.

2.1.6.1.2.4 사용자는 로그인한 디바이스에 대해 이후 사용 시 자동 로그인 될 수 있다.

2.1.6.1.2.5 사용자는 로그인 후, ‘그림책 만들기’, ‘내가 만든 그림책’, ‘내 서재’, ‘마이페이지’ 버튼이 포함된 화면을 볼 수 있다.

2.1.6.1.2.6 사용자는 ‘그림책 만들기’ 버튼을 통해 그림책 만들기 기능을 사용한다.

2.1.6.1.2.7 사용자는 시스템에 로그아웃 할 수 있다.

2.1.6.1.2.8 사용자는 회원탈퇴가 가능하다.

2.1.6.2 스토리 생성 기능

2.1.6.2.1 사용자는 스토리 입력란에 그림책으로 제작하고 싶은 스토리를 적을 수 있다.
2.1.6.2.2 스토리 입력란은 최소 100자에서 최대 1000자까지 입력할 수 있다.
2.1.6.2.3 스토리 생성 기능은 사용자가 '스토리 나누기' 버튼을 클릭하여 화면이 전환될 경우 종료한다.

2.1.6.3 맞춤법 교정 기능

2.1.6.3.1 사용자는 스토리 입력란에 작성한 스토리의 맞춤법을 교정받을 수 있다.
2.1.6.3.2 맞춤법 교정 기능은 스토리 생성창에서 '맞춤법 교정' 버튼을 통해 실행한다.
2.1.6.3.3 앱은 스토리를 복사해 NLP API로 전달한다.
2.1.6.3.4 앱은 NLP API로부터 전달 받은 맞춤법 결과를 화면에 출력한다.
2.1.6.3.5 앱은 맞춤법이 틀린 글자를 빨간색으로 표시한다.

2.1.6.4 스토리 나누기 기능

2.1.6.4.1 앱은 생성된 스토리를 복사해 chat GPT로 전달한다.
2.1.6.4.2 chat GPT는 생성된 스토리를 분석한 후, 내용에 맞게 문단을 나눈다.
2.1.6.4.3 chat GPT는 각 문단을 순차적으로 앱에 전송한다.
2.1.6.4.4 앱은 chat GPT로부터 전송받은 각 문단을 문단 별로 화면에 출력한다.
2.1.6.4.5 스토리 나누기 기능은 사용자가 '스토리 요약하기' 버튼을 클릭하여 화면이 전환될 경우 종료한다.

2.1.6.5 스토리 요약 기능

2.1.6.5.1 앱은 각 문단을 복사해 chat GPT로 전달한다.
2.1.6.5.2 chat GPT는 각 문단을 한 문장으로 요약하여 앱에 전송한다.
2.1.6.5.3 앱은 chat GPT로부터 전송받은 요약된 문장을 화면에 출력한다.
2.1.6.5.4 스토리 요약 기능은 사용자가 '적합한 그림 생성하기' 버튼을 클릭하여 화면이 전환될 경우 종료한다.

2.1.6.6 그림 생성 기능

2.1.6.6.1 앱은 chat GPT에 그림 생성 프롬프트를 전송한다.

2.1.6.6.2 그림 생성 기능은 각각의 요약된 문장을 NPL를 사용하여 키워드를 추출한다.
2.1.6.6.3 앱은 추출한 키워드를 chat GPT에 전달한다.
2.1.6.6.4 chat GPT는 각각의 문장에 적합한 그림을 생성하여 앱에 전달한다.
2.1.6.6.5 앱은 chat GPT로부터 받은 그림을 화면에 각각의 요약된 문장과 함께 출력한다.
2.1.6.6.6 사용자는 '그림 다시 생성하기' 버튼을 클릭하여 원하는 문장의 그림을 다시 생성할 수 있다.
2.1.6.6.7 그림 생성 기능은 사용자가 '그림책 편집하기' 버튼을 클릭하여 화면이 전환될 경우 종료한다.

2.1.6.7 레이아웃 및 디자인 편집 기능

2.1.6.7.1 앱은 사용자가 그림책을 생성할 때, 글자와 그림의 위치를 결정할 수 있는 기능을 제공한다.
2.1.6.7.2 사용자는 그림책에 들어가는 스토리를 왼쪽, 가운데, 오른쪽으로 정렬할 수 있다.
2.1.6.7.3 사용자는 그림책의 쪽 번호를 매길 수 있다.
2.1.6.7.4 사용자는 그림책의 여러 그림 중 각각의 페이지에 어울리는 그림을 선택하여 배치할 수 있다.
2.1.6.7.5 사용자는 원하는 그림이 존재할 시 직접 사진을 추가할 수 있다.
2.1.6.7.6 사용자는 스토리의 글자 크기, 글꼴, 색, 두께를 변경할 수 있다.
2.1.6.7.7 사용자는 그림책의 여러 그림 중 표지로 생성될 대표 그림을 선택할 수 있다.
2.1.6.7.8 사용자는 그림책의 제목을 입력할 수 있다.

2.1.6.8 저장 및 공유 기능

2.1.6.8.1 앱은 사용자가 그림책을 저장할 수 있도록 한다.
2.1.6.8.2 사용자는 '내 서재'에서 생성된 그림책을 공유할 수 있다.
2.1.6.8.3 앱은 사용자가 생성된 그림책을 공유 할 시, 해당 그림책의 URL을 생성한다.

2.2 비기능적 요구사항 분석

2.2.1 성능 요구사항

2.2.1.1 앱은 그림 생성 시, 생성 시간은 10초 이내로 한다.
2.2.1.2 앱은 그림 생성 시간 외에, 사용자 요청에 대한 응답 시간을 2초 이내로 한다.
2.2.1.3 앱은 그림과 그림책 데이터를 효과적으로 처리하고 관리할 수 있어야 한다.

2.2.2 신뢰성 요구사항

2.2.2.1 앱은 사용자가 작성한 글에 맞는 적절한 그림을 제공해야한다.

2.2.2.2 앱이 다운될 경우 빠른 시일 내에 복구 과정을 거쳐야 한다.

2.2.3 부하 및 스트레스 요구사항

2.2.3.1 앱은 한 번에 100명 이상의 사용자가 동시에 그림책 작성을 시도하는 상황에서도 안정적으로 작동해야 한다.

2.2.3.2 부하 테스트 결과, 100명 이상의 사용자가 동시에 그림책을 만들더라도 응답 시간은 3초 이내로 유지되어야 한다.

2.2.3.3 스트레스 테스트 결과, 200명 이상의 사용자가 1분 동안 동시에 그림책을 생성해도 시스템은 중단 없이 작동되어야 한다.

2.2.4 보안 요구사항

2.2.4.1 사용자의 개인 정보는 관리자만 접근할 수 있도록 한다.

2.2.4.2 사용자가 공유하지 않은 그림책은 외부에 유출되지 않도록 보호해야한다.

2.2.4.3 사용자가 공유한 그림책은 오직 읽기만 가능하도록 한다.

2.2.5 운영**(**용량**)** 요구사항

2.2.5.1 앱은 24시간 작동되어야 한다.

2.2.5.2 앱은 안드로이드 운영체제에서 작동해야 한다.

2.2.5.3 맞춤법 교정 기능과 그림 생성 기능에는 카카오 클라우드 NLP 를 사용해야한다.

2.2.5.4 Chat gpt는 GPT-3.5버전을 사용한다.

2.2.5.5 문단 나누기 기능에서 나눠진 문단들은 레이아웃 및 디자인 편집 기능에서 사용할 수 있도록 내부 데이터베이스에 임시로 저장되어야 한다.

2.2.6 사용성 요구사항

2.2.6.1 메인 화면에서는 '그림책 만들기', '내가 만든 그림책', '내 서재', '마이페이지' 버튼이 각각 명확하게 구분되어 있어야 한다.

2.2.6.2 사용자는 화면에 나타난 버튼을 클릭하여 원하는 기능에 빠르게 접근할 수 있어야 합니다.

2.2.7 호환성 요구사항

2.2.7.1 앱은 다양한 안드로이드 디바이스에서 호환되어야 한다.

2.2.7.2 화면 크기, 해상도, 버전 등 다양한 조건에서도 일관된 성능을 제공해야 한다.

2.2.8 안정성 및 가용성 요구사항

2.2.8.1 예기치 못한 오류 발생 시, 해당 오류에 대한 사용자에게 명확한 알림이 표시되어야 한다. 사용자는 오류 발생 시, 안정적인 상태로 앱을 계속 사용할 수 있어야 한다.

2.2.8.2 시스템은 24시간 작동해야 하며, 일부 서버의 다운 시에도 다른 서버가 사용자 요청을 처리할 수 있어야 한다.

2.2.9 확장성 요구사항

2.2.9.1 사용자 수의 증가에 따라 자동으로 서버를 확장하여 추가 부하를 분산해야 한다. 동시 접속자 수가 급증할 경우, 자동으로 서버를 확장하여 응답 시간을 유지해야 한다.

2.2.10 유지보수성 요구사항

2.2.10.1 앱 사용에 문제가 생겼을 경우 신속한 해결을 최우선으로

한다.

2.2.10.2 요구사항 변경이 발생할 시 이를 받아들이고 수정한다.

2.2.10.3 개발자는 앱에 추가적인 기능이 필요하다고 생각될 시 이를 개발해 앱에 반영한다.

2.2.11 설치 요구사항

2.2.11.1 사용자는 안드로이드 운영체제 버전 10.0 이상을 탑재한 모바일 기기에 앱을 설치해야 한다.

2.2.11.2 앱은 최소 50MB의 저장 공간을 요구하며, 설치 전에 사용자에게 해당 정보를 알려야 한다.

2.2.12 복구 요구사항

2.2.12.1 앱은 다운타임이 발생했을 때 자동으로 시스템을 복구할 수 있어야 한다. 시스템 다운 시, 최대 30분 이내에 복구되어야 한다.

2.2.13 법적 규정 및 환경 요구사항

(해당 없음)

2.2.14 위험분석 및 제약 요구사항

2.2.14.1 시스템은 사용자의 개인정보를 안전하게 저장하고 전송해야 하며, 보안 위험에 대한 분석 및 대응 방안이 포함되어야 한다.

2.2.14.2 앱은 안정적으로 작동하기 위해 사용자의 디바이스에서 최소 2GB RAM 및 1GHz 이상의 프로세서 속도를 요구한다.

2.2.15 문서 산출물 요구사항

2.2.15.1 프로젝트 계획서에는 앱의 개발 일정, 예산, 인력 할당 등이 명시되어야 한다.

2.2.15.2 앱의 요구사항에 대한 명세가 상세히 기록되어야 하며, 변경 시 이를 업데이트해야 한다.

3. 외부 인터페이스 요구사항

3.1 하드웨어 인터페이스 요구사항(Hardware interface requirement)

- HIR-001 앱은 그림 및 그림책 데이터를 안전하게 저장할 수 있는 충분한 데이터베이스를 필요로 한다.
- HIR-002 사용자는 앱을 작동시킬 수 있는 Android 체제의 하드웨어를 사용해야 한다.

3.2 소프트웨어 인터페이스 요구사항(Software interface requirement)

- SIR-001 앱은 데이터베이스와 24시간 연동되어야 한다.
- SIR-002 앱은 외부 API와의 통신을 위해 RESTful API를 사용하여 JSON형식의 데이터를 주고 받을수 있어야 한다.

3.3 통신 인터페이스 요구사항(Communication interface requirement)

- CIR-001 클라이언트와 서버 간의 통신은 HTTPS를 사용하며, 데이터 전송은 TLS프로토콜을 통해 암호화 되어야 한다.
- CIR-002 서버는 발생 가능한 오류에 대해 오류 메시지와 함께 상태코드를 클라이언트에게 전송해야 한다.

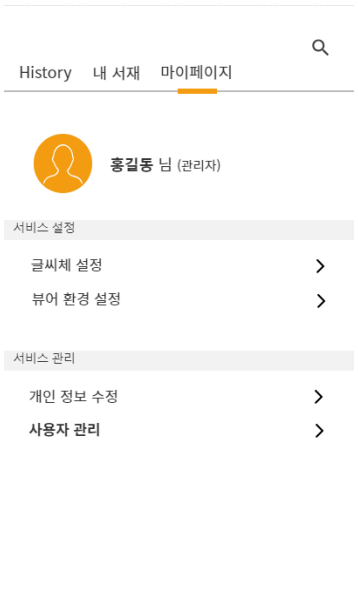
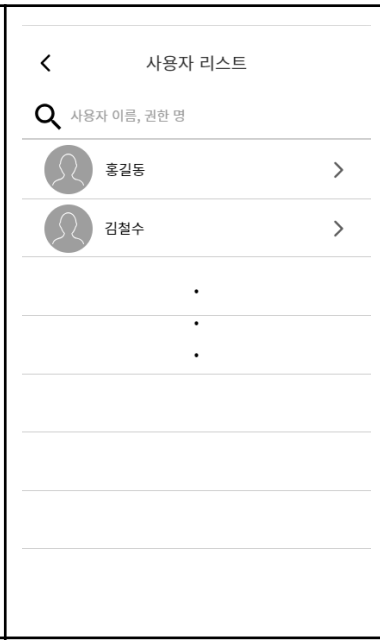
4. 사용자 인터페이스 설계

4.1 사용자 인터페이스 요구사항(User interface requirement)

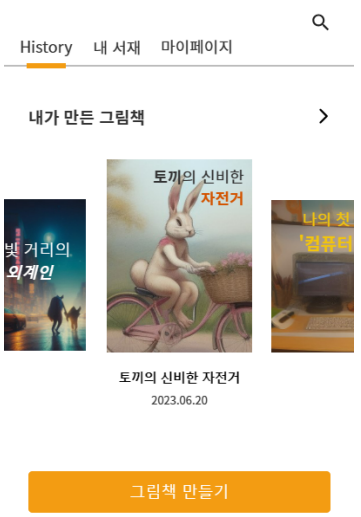
- UIR-001 안드로이드 운영체제에서 작동해야 한다.
- UIR-002 시스템은 사용자가 '그림책 만들기' 과정 도중에 프로그램을 종료하려 할 때, 경고 메세지를 띄워야한다.
- UIR-003 시스템은 사용자가 만들기 도중 프로그램을 종료하면 진행 내역을 저장해야한다.
- UIR-004 시스템은 그림책 만들기 전 전반적인 그림책 만들기의 설명과 진행 방법을 대상자로 하여금 알 수 있도록 출력해주어야한다.
- UIR-005 사용자는 로그인 후, '그림책 만들기', '내가 만든 그림책', '내 서재', 마이페이지 버튼이 포함된 화면을 볼 수 있다.

4.2 사용자 인터페이스 설계

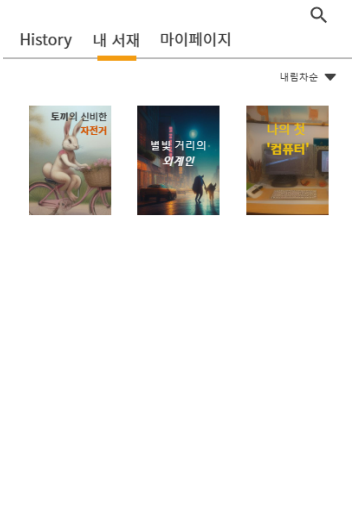
4.2.1 관리자용 사용자 인터페이스 설계

		
관리자 마이페이지	사용자 리스트 관리	

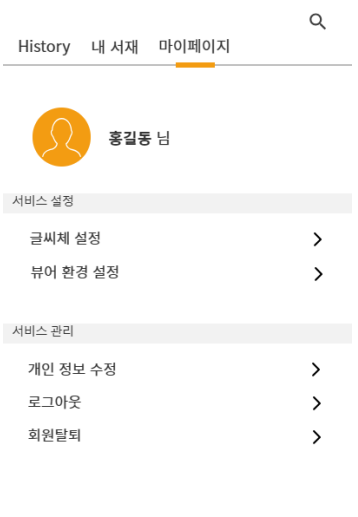
4.2.2 사용자용 사용자 인터페이스 설계



메인 화면**(history)**

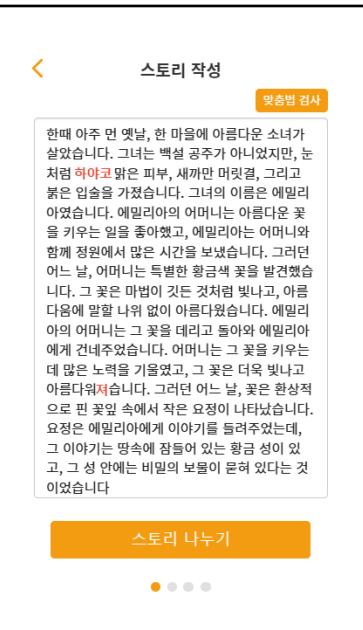


내 서재

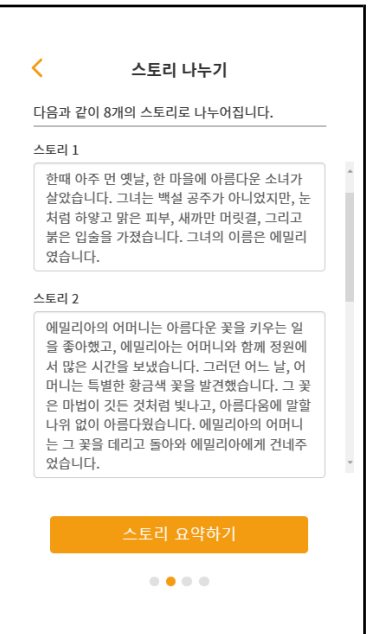


마이페이지

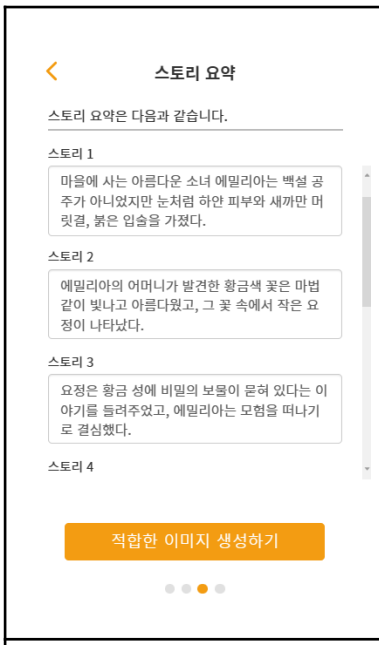
스토리 작성



맞춤법 자동 교정

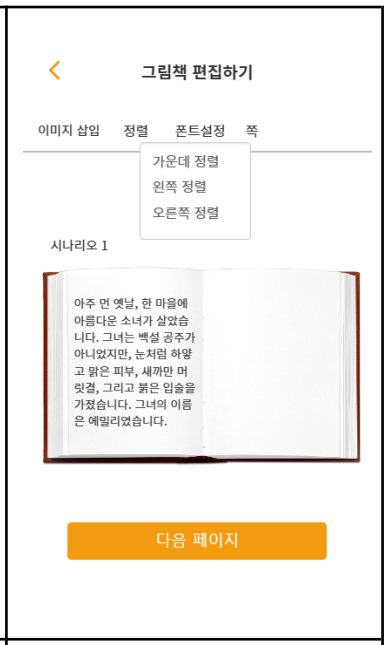


스토리 나누기

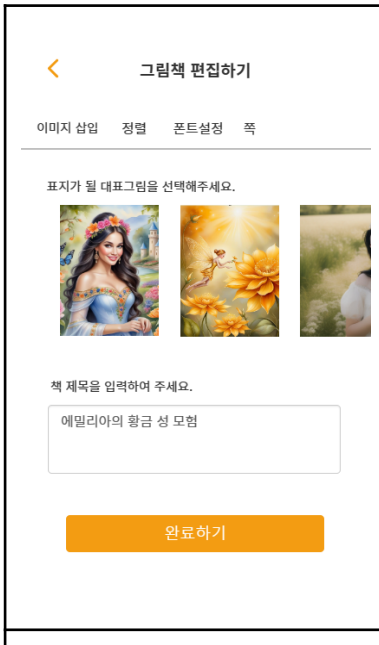


스토리 요약

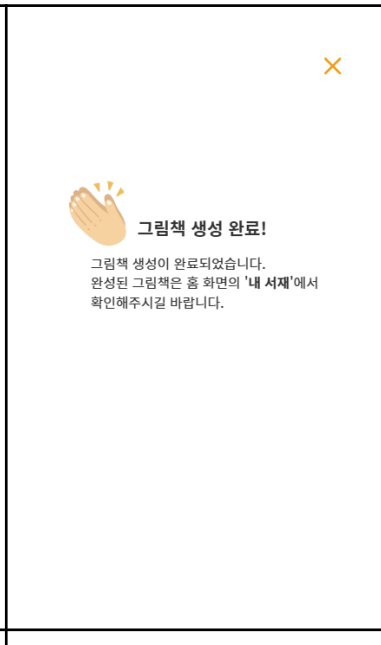
문단 별 그림 생성



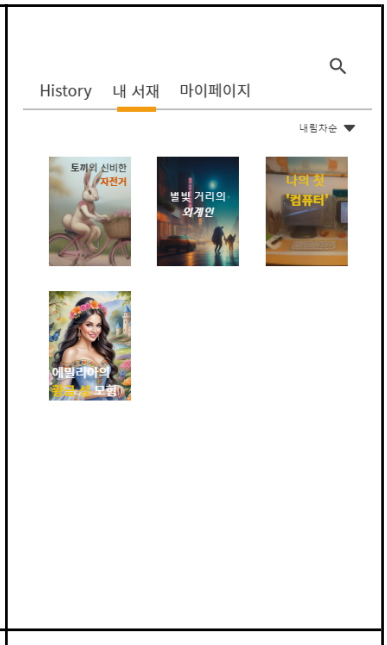
그림책 만들기(편집)



표지, 제목 설정



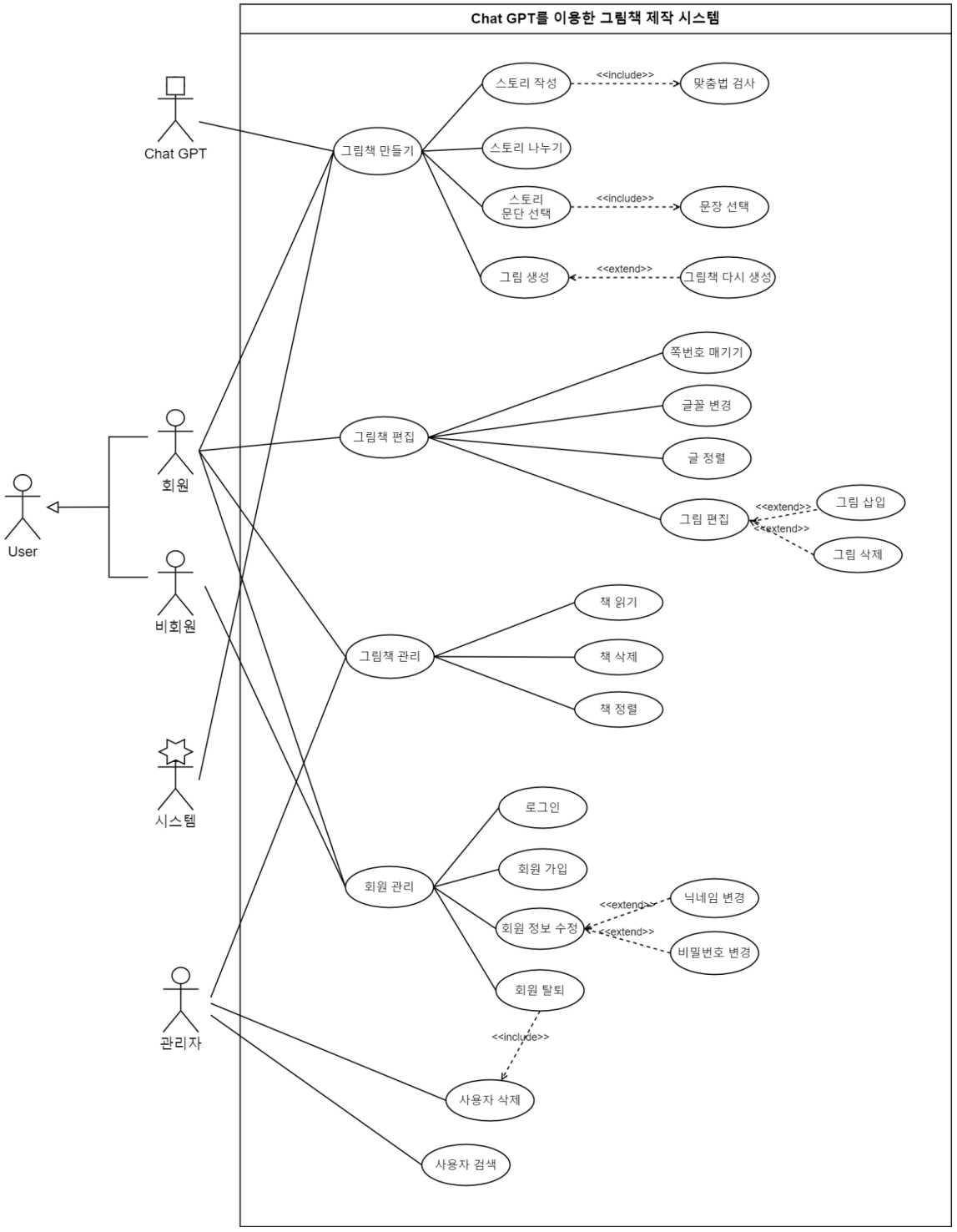
그림책 생성 완료



내 서재 확인

5. 기능적 모델링

5.1 사용사례 다이어그램



5.1.1 액터 리스트

식별자	액터명	역할
ACT-0	User	앱이 제공하는 기능을 사용하여 그림책을 제작하는 사람
ACT-1	System	그림책 생성 기능과 서재 조회 기능을 사용자에게 제공
ACT-2	관리자	User들의 정보를 관리하는 사람
ACT-3	ChatGPT	그림책 생성을 도와주는 API

5.2 사용사례 명세(사용 시나리오)

5.2.1 Use Case : 스토리 작성

Use Case Name	스토리 작성	**ID**	UC-001	**ImportanceLevel**	중
Primary Actor	회원, 시스템	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 그림책의 스토리를 입력한다. 시스템은 회원이 입력한 스토리의 맞춤법 검사 기능을 진행한다.				
Brief Description	이 use case는 회원이 그림책의 스토리를 작성하는 과정을 보여준다.				
Trigger	회원은 '그림책 만들기' 버튼을 통해 스토리 작성 화면으로 접근한다.				
Relationships	Association: User Include :usecase[맞춤법 검사]				
Normal Flow of Event (Normal Scenario)					
1. 회원은 앱의 홈 화면에서 "그림책 만들기" 버튼을 클릭한다. 2. 시스템은 스토리 입력란을 화면에 표시한다. 3. 회원은 입력란에 생성할 그림책의 스토리를 작성한다. 4. 회원은 작성한 스토리를 검토하기 위해 상단의 맞춤법 검사 기능 버튼을 누른다. 4.1 시스템은 스토리를 맞춤법 검사하고, 오류가 있다면 해당 부분을 강조하여 표시한다. 4.2 회원은 필요에 따라 맞춤법 검사 결과를 확인하고 수정한다. 4.3 수정이 완료되면, 회원은 다시 원래의 흐름으로 돌아간다. 5. 회원은 스토리 작성이 완료되면 문단 나누기 기능 버튼을 선택할 수 있다. 5.1 시스템은 스토리를 문단으로 나누는 기능을 제공한다. 5.2 회원은 이를 활용하여 스토리의 구조를 조정한다. 5.3 문단 나누기가 완료되면, 회원은 다시 원래의 흐름으로 돌아간다. 6. 시스템은 회원이 작성한 그림책을 저장한다. 7. 시스템이 회원에게 그림책 생성 성공 메시지를 표시한다.					
Sub_Flow					

Alternative / Exception Flow

5.2.2 Use Case : 스토리 문단 선택

Use Case Name	스토리 문단 선택	ID	UC-002	ImportanceLevel	상
Primary Actor	회원, 시스템	Use Case Type		Detail, Essential	
Stakeholders	회원은 그림책 스토리의 문단을 선택해 나눈 후, 문장을 선택해 시스템에 프롬프트 형태로 전달할 수 있다.				
Brief Description	이 use case는 회원이 스토리의 문단을 선택해 나누고, 이후 문단별로 그림을 생성할 대표 문장을 선택해 시스템에게 프롬프트 형태로 전달하는 과정을 보여준다.				
Trigger	회원이 ‘그림책 만들기’ 버튼을 통해 스토리 작성 화면으로 접근한다.				
Relationships	Association: 회원 Include :usecase[문장 선택]				
Normal Flow of Event (Normal Scenario)					
1. 회원은 앱에서 작성 중인 그림책의 스토리를 화면에 표시한다. 2. 회원은 스토리에서 나누고 싶은 문단을 선택한다. 2.1 회원이 문단을 선택하지 않으면, 시스템은 오류 메시지를 표시하고 다시 문단 선택을 요청한다. 2.2 회원은 올바른 문단을 선택할 때까지 반복한다. 3. 시스템은 회원이 선택한 문단을 화면에 표시한다. 4. 회원은 문단 내에서 특정 문장을 선택하여 그림으로 나타내고 싶은 부분을 선택한다. 4.1 회원이 문장을 선택하지 않으면, 시스템은 오류 메시지를 표시하고 다시 문장 선택을 요청한다.					

4.2 회원은 올바른 문장을 선택할 때까지 반복한다. 5. 회원은 선택한 문장에 대한 그림을 추가하기 위해 시스템에게 프롬프트 형태로 정보를 전달한다.
Sub_Flow
Alternative / Exception Flow
2.a 회원이 문단을 선택하지 않으면, 문단을 나눌 수 없다. 4.a 회원이 문장을 선택하지 않으면, 시스템에게 그림을 생성하는 프롬프트를 전달할 수 없다.

5.2.3 Use Case : 스토리 나누기

Use Case Name	스토리 나누기	**ID**	UC-003	**ImportanceLevel**	상
Primary Actor	회원, 시스템, chatGPT	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 그림책에 들어갈 스토리를 작성하고 문단을 나눈다.				
Brief Description	이 use case는 회원이 Chat gpt를 통해 그림책에 들어갈 스토리를 나누는 과정을 보여준다.				
Trigger	회원이 그림책 만들기에 접근한다.				
Relationships	Association: 회원, Chat gpt				
Normal Flow of Event (Normal Scenario)					

1. 회원은 스토리를 작성한 후 "스토리 나누기" 버튼을 클릭한다. 2. 시스템은 작성된 스토리를 복사하여 Chat GPT에게 전달한다. 3. Chat GPT는 전달받은 스토리를 분석하여 적절한 문단으로 나눈다. 4. Chat GPT는 나눈 문단을 하나씩 시스템에 전달한다. 5. 시스템은 Chat GPT로부터 받은 각 문단을 데이터베이스(DB)에 저장한다. 5.1 저장된 데이터는 회원, 스토리 ID, 문단 순서 등과 연결돼 정리된다. 5.2 동시에 시스템은 화면에 사용자가 작성한 스토리의 각 문단을 출력하며, 사용자에게 확인할 수 있는 인터페이스를 제공한다.
Sub_Flow
Alternative / Exception Flow

5.2.4 Use Case : 그림 생성

Use Case Name	그림 생성	**ID**	UC-004	**ImportanceLevel**	상
Primary Actor	회원, 시스템, Chat gpt	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 그림책에 들어갈 그림을 생성한다.				
Brief Description	이 use case는 회원이 Chat gpt를 사용해 그림을 생성하는 과정을 보여준다.				
Trigger	회원이 그림책 만들기에 접근한다.				
Relationships	Association: 회원, 시스템				

	Extend: usecase[그림 다시 생성하기]
Normal Flow of Event (Normal Scenario)	
1. 회원은 그림책 만들기에 들어가 작업을 수행한다. 2. 회원은 그림책의 내용을 바탕으로 원하는 그림을 요청한다. 3. 시스템은 문장 요약단계에서 임시로 저장했던 프롬프트 형식의 문장을 Chat gpt에게 전송한다. 4. Chat gpt는 프롬프트에 맞게 그림을 생성하고 생성된 그림을 다시 시스템에게 전송한다. 5. 시스템은 전송 받은 그림을 차례로 화면에 출력한다. 6. 회원은 원하는 그림이 아닐 시, Chat gpt에게 그림을 다시 생성하도록 요청한다. 6.1 회원이 그림 다시 생성하기 버튼을 누를 시, 시스템은 다시 Chat gpt에게 그림 생성을 요청한다. 6.2 회원이 그림 다시 생성하기 버튼을 누르지 않는다면, 그림책 편집 화면으로 넘어간다.	
Sub_Flow	
4.1s 그림을 다시 생성할 경우 Chat gpt는 새로운 그림을 사용자에게 제공한다.	
Alternative / Exception Flow	

5.2.5 Use Case : 그림책 편집

Use Case Name	그림책 편집	**ID**	UC-005	**ImportanceLevel**	중
Primary Actor	사용자, 시스템	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 작성한 그림책을 편집한다.				
Brief Description	이 use case는 회원이 그림책을 편집하는 과정을 보여준다.				

Trigger	회원이 그림책 만들기에 접근한다.
Relationships	Association: ChatGPT, 회원, 시스템
Normal Flow of Event (Normal Scenario)	
1. 회원은 그림생성이 끝나면, 그림책 편집 버튼을 누른다. 2. 그림책 편집 버튼이 눌리면, 시스템은 생성된 그림과, 문단을 가져온다. 3. 가져온 문단은 그림책 페이지의 홀수 쪽에 삽입된다. 4. 가져온 그림은 그림책 페이지의 짝수 쪽에 삽입된다. 5. 사용자는 그림책 쪽번호를 매길 수 있다. 5.1 사용자가 쪽번호 매기기를 선택하면 그림책은 하단에 현재 페이지를 표시한다. 6. 사용자는 글꼴 설정을 할 수 있다. 6.1 사용자가 변경할 글꼴을 선택하면, 시스템은 그 글꼴의 형식에 맞게 글자간 간격. 줄 간격을 조정한다. 7. 사용자는 정렬기능을 사용할 수 있다. 7.1 사용자는 왼쪽, 오른쪽, 가운데 정렬을 선택할 수 있다. 7.1.1 사용자가 왼쪽 정렬을 선택하면 시스템은 가장 왼쪽 상단의 글자를 기준으로 모든 글자 정렬을 맞춘다. 7.1.2 사용자가 오른쪽 정렬을 선택하면 시스템은 가장 오른쪽 상단의 글자를 기준으로 모든 글자 정렬을 맞춘다. 7.1.2 사용자가 가운데 정렬을 선택하면 시스템은 가장 가운데 상단의 글자를 기준으로 모든 글자 정렬을 맞춘다.	
Sub_Flow	
Alternative / Exception Flow	

5.2.6 그림책 관리

Use Case Name	그림책 관리	ID	UC-006	ImportanceLevel	하
Primary Actor	회원, 관리자	Use Case Type		Detail, Essential	
Stakeholders	회원은 작성한 그림책 읽기, 책 삭제, 책 정렬을 할 수 있다.				
Brief Description	이 use case는 회원이 그림책을 관리하는 과정을 보여준다.				
Trigger	회원이 내 서재에 접근한다.				
Relationships	Association: 회원, 시스템, 관리자				

Normal Flow of Event (Normal Scenario)

1. 회원은 메뉴에서 내 서재를 클릭하여 내 서재에 접근한다.
   1.1 회원은 내 서제에서 삭제하고자 하는 그림책을 선택한다.
2. 회원은 그림책들을 삭제, 읽기, 정렬할 수 있다.
   [삭제 서브 플로우]
   2.1 회원은 내 서재에서 삭제하고자 하는 그림책을 선택하고 삭제할 수 있다.
   2.2 삭제된 그림책은 더 이상 내 서재에서 확인할 수 없다.
   [그림책 읽기 서브 플로우]
   2.3 회원은 내 서재에서 읽고자 하는 그림책을 선택한다.
   2.4 시스템은 선택한 그림책의 내용을 불러와 화면에 출력한다.
   [그림책 정렬 서브 플로우]
   2.5 회원은 내 서재에서 그림책들을 정렬할 수 있다.
   2.6 회원은 정렬 기준을 선택한다(예: 날짜순, 이름순).
   2.7 시스템은 선택한 정렬 기준에 따라 그림책을 정렬한다.
3. 회원이 그림책을 삭제한다면, 시스템은 해당 그림책을 삭제한다.
4. 회원이 그림책을 읽는다면, 시스템은 해당 그림책의 내용을 불러와 화면에 출력한다.
5. 회원이 그림책들을 정렬하면, 시스템은 그림책을 날짜순, 이름순으로 정렬한다.

Sub_Flow

Alternative / Exception Flow

5.2.7 로그인

Use Case Name	로그인	**ID**	UC-007	**ImportanceLevel**	하
Primary Actor	사용자, 시스템	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 시스템에 로그인 한다.				
Brief Description	이 use case는 회원이 로그인하는 로직을 보여준다.				
Trigger	회원이 로그인을 시도한다.				
Relationships	Association: ChatGPT, 회원, 시스템				
Normal Flow of Event (Normal Scenario)					
1. 회원은 로그인 창에 아이디와 비밀번호를 입력한다. 2. 시스템은 회원이 입력한 아이디와 비밀번호가 DB에 저장된 회원 번호와 일치하는지 확인한다. 2.1 입력된 아이디가 일치하지 않으면, 알림메시지를 보내고 다시 입력 받도록 한다. 2.2 입력된 비밀번호가 일치하지 않으면, 알림메시지를 보내고 다시 입력 받도록 한다.					

Sub_Flow
Alternative / Exception Flow
2e. 로그인이 5번 이상 실패하면, 아이디/비밀번호 찾기를 시도하도록 한다.

5.2.8 Use Case : 회원 정보 수정

Use Case Name	회원 정보 수정	**ID**	UC-008	**ImportanceLevel**	하
Primary Actor	회원	**Use Case Type**		Detail, Essential	
Stakeholders	회원은 그림책에 들어갈 그림을 생성한다.				
Brief Description	이 use case는 회원이 Chat gpt를 사용해 그림을 생성하는 과정을 보여준다.				
Trigger	회원이 그림책 만들기에 접근한다.				
Relationships	Association: 회원 Extend: usecase[닉네임 변경, 비밀번호 변경]				
Normal Flow of Event (Normal Scenario)					
1. 회원은 웹 사이트에 접속하여 회원 관리 페이지에 접근한다. 1.1 회원은 로그인 상태인지 확인한다. 1.2 로그인 상태가 아니라면, 로그인 페이지로 이동하여 로그인을 진행한다. 2. 시스템은 회원 가입, 회원 정보 수정, 회원 탈퇴 등의 기능을 가진 버튼을 화면에 출력한다.					

3. 회원은 회원 정보 수정 버튼을 선택한다. 3.1 회원은 마우스 혹은 터치를 이용해 '회원 정보 수정' 버튼을 클릭한다. 4.시스템은 회원 정보 수정 창을 띄우고 회원의 정보를 조회해 화면에 보여준다. 4.1 시스템은 데이터베이스에서 해당 회원의 정보를 조회한다. 4.2 조회된 정보는 이름, 이메일, 전화번호 등 개인정보와, 비밀번호를 제외한 회원 정보가 화면에 표시된다. 5. 회원은 새로운 회원 정보를 입력해 수정한다. 5.1 회원은 개인정보 중 변경하고 싶은 항목을 찾아 입력란에 새로운 정보를 입력한다. 5.2 모든 변경이 완료되면 '정보 수정' 버튼을 클릭한다. 6. 시스템은 회원이 입력한 정보를 저장한다. 6.1 시스템은 회원이 입력한 새로운 정보를 데이터베이스에 저장하고, 성공적으로 저장되었다는 메시지를 화면에 출력한다. 6.2 만약 정보 저장 중 문제가 발생하면, 해당 문제를 알리는 메시지를 화면에 출력한다.
Sub_Flow
Alternative / Exception Flow
5e. 회원이 입력한 정보에 변경사항이 없다면, 다시 입력할 것을 요청한다.

5.2.9 Use Case : 회원 탈퇴

Use Case Name	회원 탈퇴	ID	UC-009	ImportanceLevel	하
Primary Actor	회원	Use Case Type		Detail, Essential	
Stakeholders	회원은 그림책 만들기 앱을 탈퇴한다.				
Brief Description	이 use case는 회원이 그림책 만들기 앱을 탈퇴하는 과정을 보여준다.				
Trigger	회원이 회원 관리에 접근한다.				
Relationships	Association: 회원 Include: usecase[사용자 삭제]				

Normal Flow of Event (Normal Scenario)

1. 회원은 웹 사이트에 접속하여 회원 관리 페이지에 접근한다.
   1.1 회원은 로그인 상태인지 확인한다.
   1.2 로그인 상태가 아니라면, 로그인 페이지로 이동하여 로그인을 진행한다.
2. 시스템은 회원 가입, 회원 정보 수정, 회원 탈퇴 등의 기능을 가진 버튼을 화면에 출력한다.
3. 회원은 회원 정보 수정 버튼을 선택한다.
   3.1 회원은 마우스 혹은 터치를 이용해 '회원 정보 수정' 버튼을 클릭한다.
4. 시스템은 회원 정보 수정 창을 띄우고 회원의 정보를 조회해 화면에 보여준다.
   4.1 시스템은 데이터베이스에서 해당 회원의 정보를 조회한다.
   4.2 조회된 정보는 이름, 이메일, 전화번호 등 개인정보와, 비밀번호를 제외한 회원 정보가 화면에 표시된다.
5. 회원은 새로운 회원 정보를 입력해 수정한다.
   5.1 회원은 개인정보 중 변경하고 싶은 항목을 찾아 입력란에 새로운 정보를 입력한다.
   5.2 모든 변경이 완료되면 '정보 수정' 버튼을 클릭한다.
6. 시스템은 회원이 입력한 정보를 저장한다.
   6.1 시스템은 회원이 입력한 새로운 정보를 데이터베이스에 저장하고, 성공적으로 저장되었다는 메시지를 화면에 출력한다.
   6.2 만약 정보 저장 중 문제가 발생하면, 해당 문제를 알리는 메시지를 화면에 출력한다.

Sub_Flow
Alternative / Exception Flow
3e. 회원이 “아니오” 버튼을 선택했다면 다시 이전 페이지로 돌아간다.

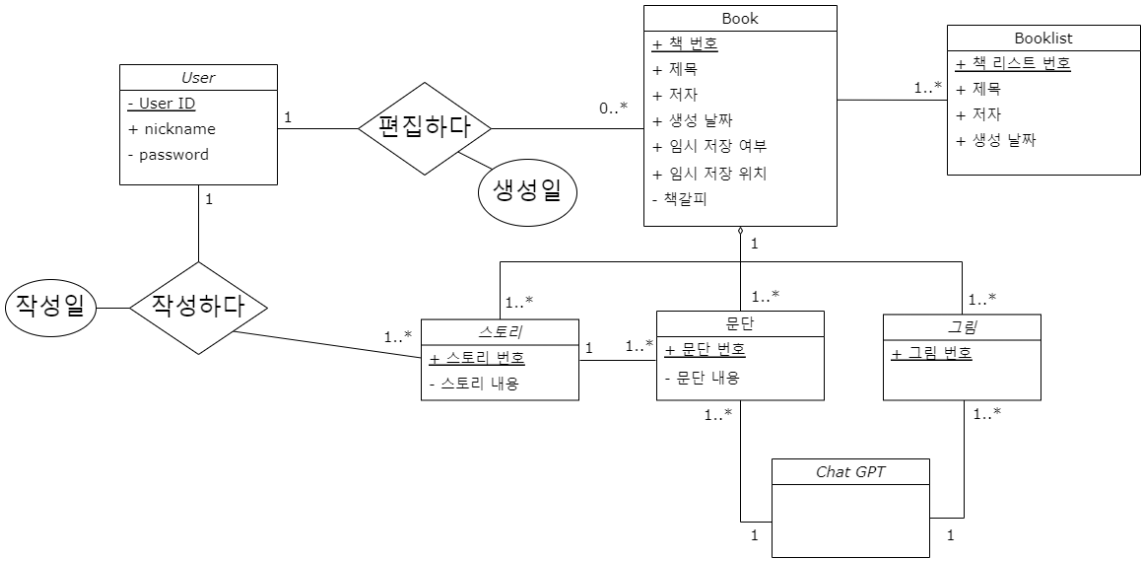
5.2.10 Use Case : 회원 가입

Use Case Name	회원 가입	**ID**	UC-010	**ImportanceLevel**	하
Primary Actor	비회원	**Use Case Type**		Detail, Essential	
Stakeholders	비회원이 그림책 만들기 앱에 새롭게 가입한다.				
Brief Description	이 use case는 회원이 그림책 만들기 앱을 가입하는 과정을 보여준다.				
Trigger	비회원이 회원 가입에 접근한다.				
Relationships	Association: 비회원				
Normal Flow of Event (Normal Scenario)					
1. 비회원이 그림책 생성 APP에 접근해 회원가입 버튼을 선택한다. 2. 시스템은 회원 가입 관련 페이지를 화면에 출력한다. 3. 비회원은 회원 가입을 위해 빈칸에 내용을 입력한다. 3.1 비회원은 이름을 필수로 작성한다. 3.2 비회원은 닉네임을 필수로 작성한다. 3.3 비회원은 비밀번호를 필수로 작성한다. 3.4 비회원은 전화 번호를 필수로 작성한다.					

3.5 비회원은 이메일을 필수로 작성한다. 3.2.1 비회원은 닉네임 중복 확인을 한다. 3.3.1 비회원은 비밀번호 다시 작성하기를 통해 비밀번호를 한번 더 입력한다. 3.2 비회원은 이외의 내용들에 대해선 선택적으로 작성한다. 4. 비회원은 회원 가입 완료 버튼을 클릭한다.
Sub_Flow
Alternative / Exception Flow
3e. 비회원이 필수 입력이 필요한 빈칸을 입력하지 않고 다음단계로 넘어가는 버튼을 눌렀다면, 필수 입력 빈칸을 빨간색으로 변경하고 입력을 요청한다. 3.2.1e. 닉네임이 중복된다면 닉네임 칸을 빨간색으로 표시하고 '중복되는 아이디가 있습니다'를 출력한다. 3.3.1e. 비회원이 비밀번호 작성 시, 요구되는 사항들중 지키지 않은 것이 있다면 비밀번호 칸을 빨간색으로 표시하고 '비밀번호를 다시 작성해주세요'를 출력한다.

6. 구조적 모델링

6.1 클래스 다이어그램



6.2 CRC cards와 클래스 명세서

6.2.1 클래스 명세서

1) class :

User, Book, Booklist, 스토리, Chat GPT, 그림, 문단

2) attributes:

- **User ID :** 사용자를 구별할 수 있는 고유 id를 의미한다.
- **nickname :** 사용자가 '저자명'으로 사용될 이름을 의미한다.
- **password** : 사용자가 로그인을 위해 사용할 비밀번호를 의미한다.
- 제목 **:** 책의 제목을 의미한다.
- 저자 **:** 책의 저자를 말한다, 앱에서 저자는 닉네임으로 작성된다.
- 생성날짜 **:** 책이 완성된 날짜를 의미한다. 이 날짜는 책의 히스토리를 구분하는데 사용된다.
- 임시 저장 여부 **:** 사용자가 책 생성도중 앱에서 나가게 되면, 앱은 생성하고 있던 책을 저장하고, 부울값을 부여한다.
- 임시 저장 위치 **:** 사용자가 책 생성 도중 중단시, 그 위치를 저장한다. 순서대로 번호를 부여하여 임시저장이 수행된 시점의 위치의 번호를 저장한다.

  (예) 스토리 작성하기 = 1, 스토리 나누기 = 2)

- 책갈피 : 사용자가 읽은 페이지의 넘버를 저장한다.
- 스토리 번호 : 스토리를 구분할 수 있는 고유 번호를 의미한다.
- 스토리 타입 : 스토리가 문단의 형태인지 프롬프트의 형태인지 스토리의 타입을 구분한다.
- 그림 번호 : 그림들을 구분할 수 있는 고유 번호를 의미한다.

6.2.2 CRC cards

6.2.2.1 Class: User

class Name : User		ID : class-01		Type: domain	
Description : 사용자 정보를 관리하기 위한 클래스				Associated Use case :8,9,10	

멤버함수	회원 정보 검색() //회원 정보 검색 함수
	회원 정보 수정() // 회원 정보 수정 함수
	회원 삭제() // 관리자가 회원을 삭제하는 함수
	회원 탈퇴() // 사용자가 회원을 탈퇴하는 함수
	회원 가입() //회원 가입하는 함수
	스토리 수정()//그림책 스토리의 내용을 수정하는 함수
	문장 선택() // 문단 중 그림으로 만들 한 문장을 선택하는 함수				

속성	속성명	자료형	크기	제약조건	비고
아이디	userID	Varchar	20	PK	아이디 값
닉네임	nickname	Varchar	50	-	닉네임 값
비밀번호	password	Varchar	50	-	비밀번호 값

6.2.2.2 Class: Book

class Name : Book		ID : class-02		Type: domain	
Description : 생성된 그림책의 정보를 가진 클래스				Associated Use case :	

멤버함수	책 읽기() // 저장된 책들을 읽는 함수
	책 삭제하기() // 책을 삭제하는 함수
	이어서 읽기() // 책갈피의 위치부터 읽을 수 있게 해주는 함수
	처음부터 읽기() // 책을 처음부터 읽게 해주는 함수
	임시 저장 여부 조회() //책 만들기 도중 종료여부를 알려주는 함수
	임시 저장 위치 확인()// 어디서 도중 종료가 되었는지 위치를 반환하는 함수
	책갈피 위치 조회() // 책갈피의 위치 반환해주는 함수				

속성	속성명	자료형	크기	제약조건	비고
책 번호	bookID	Varchar	20	PK	책 아이디 값
제목	bookName	Varchar	50	-	책 제목
저자	nickname	Varchar	50	-	닉네임 값
생성날짜	dateOfCreation	date	-		책 생성날짜
임시저장 여부	temporaryStorageStatus	boolean	2		책 임시저장 여부
임시저장 위치	temporaryStorageLocation	boolean	2		책 임시저장된 구간의 위치
책갈피	bookmark	Int	4		책 읽기를 멈춘 구간의 위치

6.2.2.3 Class: Book List

class Name : BookList		ID : class-03		Type: domain	
Description : 생성된 그림책 리스트 정보를 가진 클래스				Associated Use case :	
멤버함수					
속성	속성명	자료형	크기	제약조건	비고
책 목록번호	bookListID	Varchar	20	PK	책 목록 아이디 값
책 번호	bookID	Varchar	20	FK	책 아이디 값
제목	bookName	Varchar	50	-	책 제목 값
저자	author	Varchar	50	-	저자 이름 값

6.2.2.4 Class: 스토리

class Name : Story		ID : class-04		Type: domain	
Description : 생성된 그림책의 스토리를 가진 클래스				Associated Use case :1, 2	

멤버함수	스토리 작성() // 사용자에게 스토리 작성을 요청하는 함수
	맞춤법 검사() // 사용자가 작성한 스토리의 맞춤법
	스토리 저장() // 사용자가 작성한 스토리를 DB에 저장				

속성	속성명	자료형	크기	제약조건	비고
스토리번호	storyID	Varchar	20	PK	스토리 아이디 값
스토리 내용	story	Varchar	50	-	스토리 내용 값

6.2.2.5 Class: ChatGPT

class Name : ChatGPT		ID : class-05		Type: domain	
Description : 문단 나누기, 그림생성을 도와주는 AI				Associated Use case :3,4	
멤버함수	스토리 전달하기() // 스토리를 GPT에게 전달해주는 API 문장 전달하기() // 문장을 GPT에게 전달해주는 API				
속성	속성명	자료형	크기	제약조건	비고
				-	

6.2.2.6 Class: 그림

class Name : Picture		ID : class-06		Type: domain	
Description : 생성된 그림책의 그림 정보를 가진 클래스				Associated Use case :4, 5	
멤버함수					
속성	속성명	자료형	크기	제약조건	비고
그림번호	pictureID	Varchar	20	PK	그림 아이디 값

6.2.2.7 Class: 문단

class Name : 문단		ID : class-07		Type: domain	
Description : 나눠진 스토리를 문단 형식으로 저장하는 클래스				Associated Use case :2, 3	
멤버함수	개별 문단 검색() // 개별 문단을 검색하는 함수 문단 저장() // 나눠진 문단을 저장하는 함수				
속성	속성명	자료형	크기	제약조건	비고
문단번호	paragraphs ID	Varchar	20	PK	문단 아이디 값

6.3 Abstract Class와 interface명세

(해당 사항 없음)

6.4 그 밖의 자료구조 명세

(해당 사항 없음)

7. 데이터베이스 스키마 명세

7.1 User 스키마 테이블 명세서

데이터 항목	변수명	자료형	길이	제약사항
사용자 아이디	UserID	varchar	20	PK
닉네임	nickname	varchar	20	
비밀번호	password	varchar	20	

7.2 Book 스키마 테이블 명세서

데이터 항목	변수명	자료형	길이	제약사항
책 번호	BookID	varchar	20	PK
제목	BookName	varchar	20	
저자	BookWriter	varchar	20	
생성 날짜	BookRegisterDate	varchar	20	
임시 저장 여부	BookSavingStatus	boolean	2	
임시 저장 위치	BookSavingLocation	varchar	20	
책갈피	Bookmark	varchar	20	

7.3 스토리 스키마 테이블 명세서

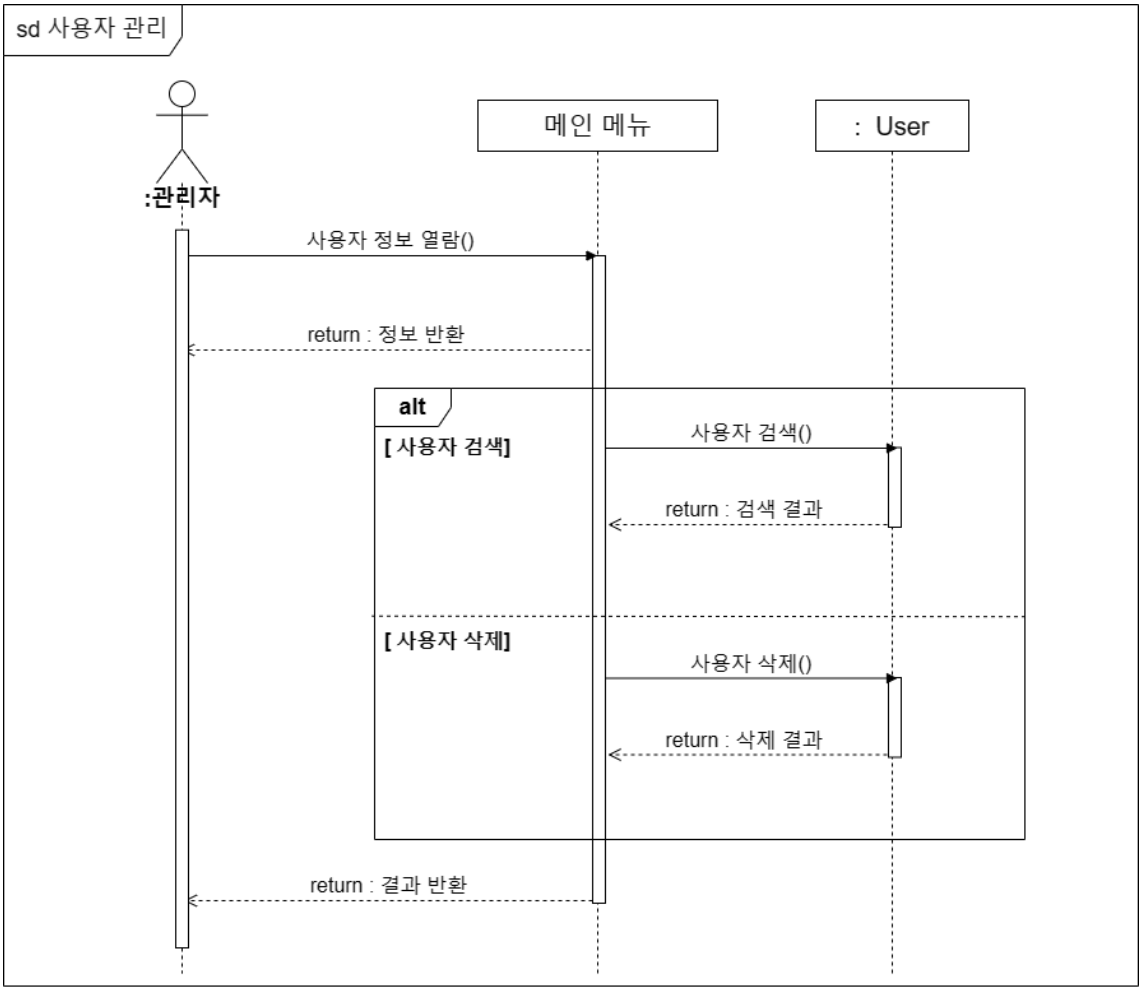
데이터 항목	변수명	자료형	길이	제약사항
스토리 번호	StoryNum	varchar	20	PK

7.4 그림 스키마 테이블 명세서

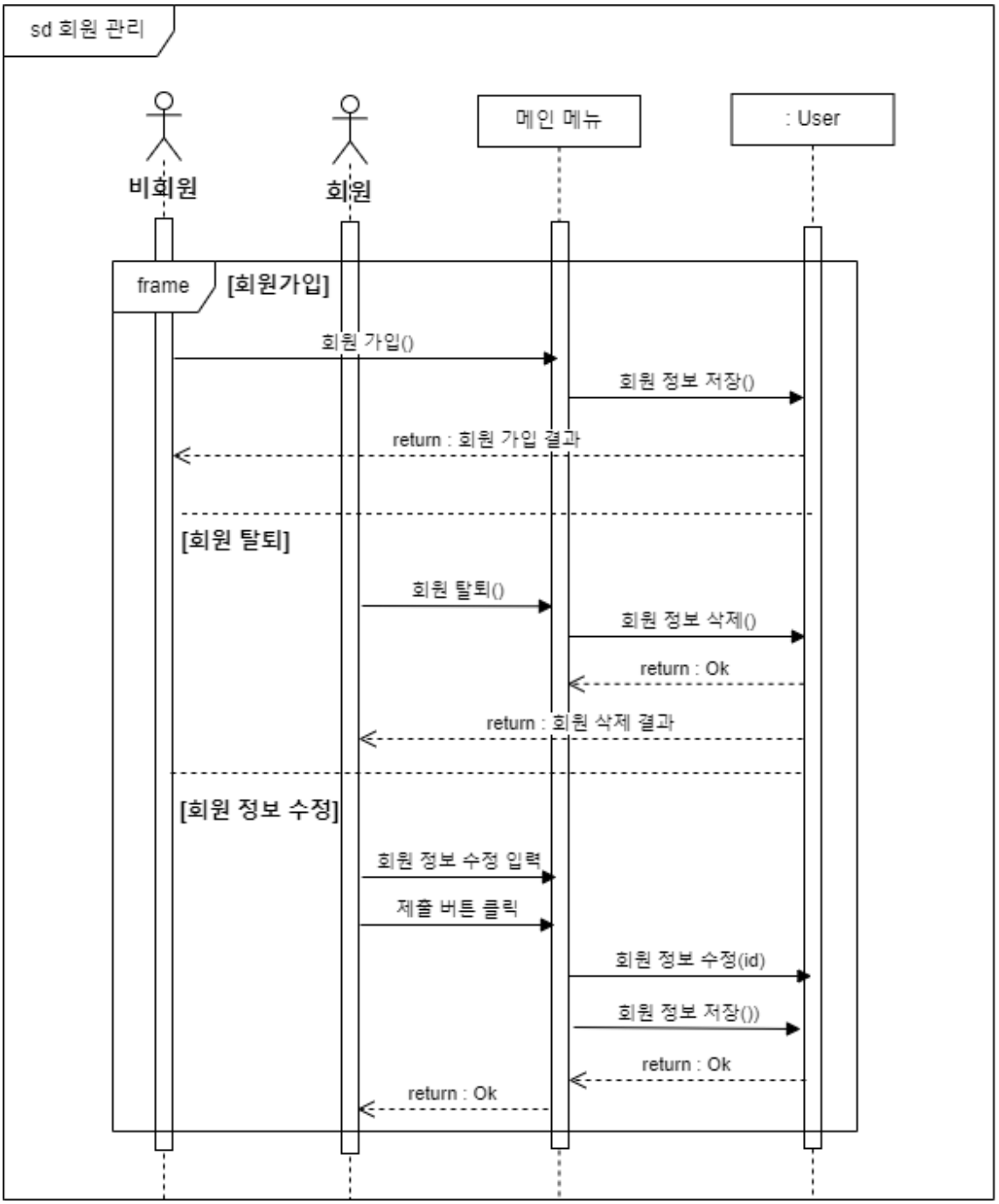
데이터 항목	변수명	자료형	길이	제약사항
그림 번호	PictureNum	varchar	20	PK

8. 행위 모델링: 사용사례에 대한 시퀀스 다이어그램

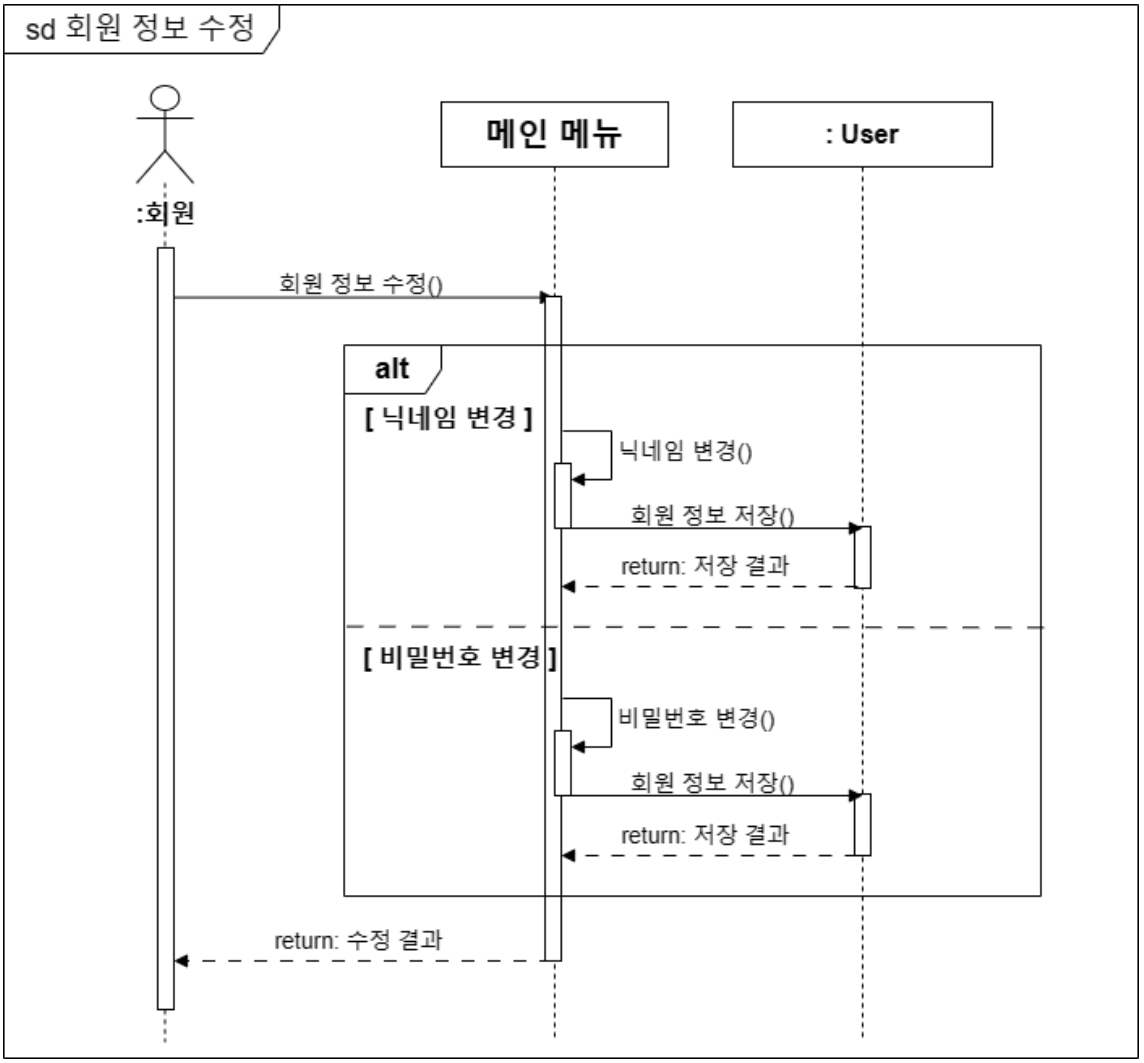
8.1 '사용자 관리(사용자 조회, 검색, 삭제)에 대한 시퀀스 다이어그램



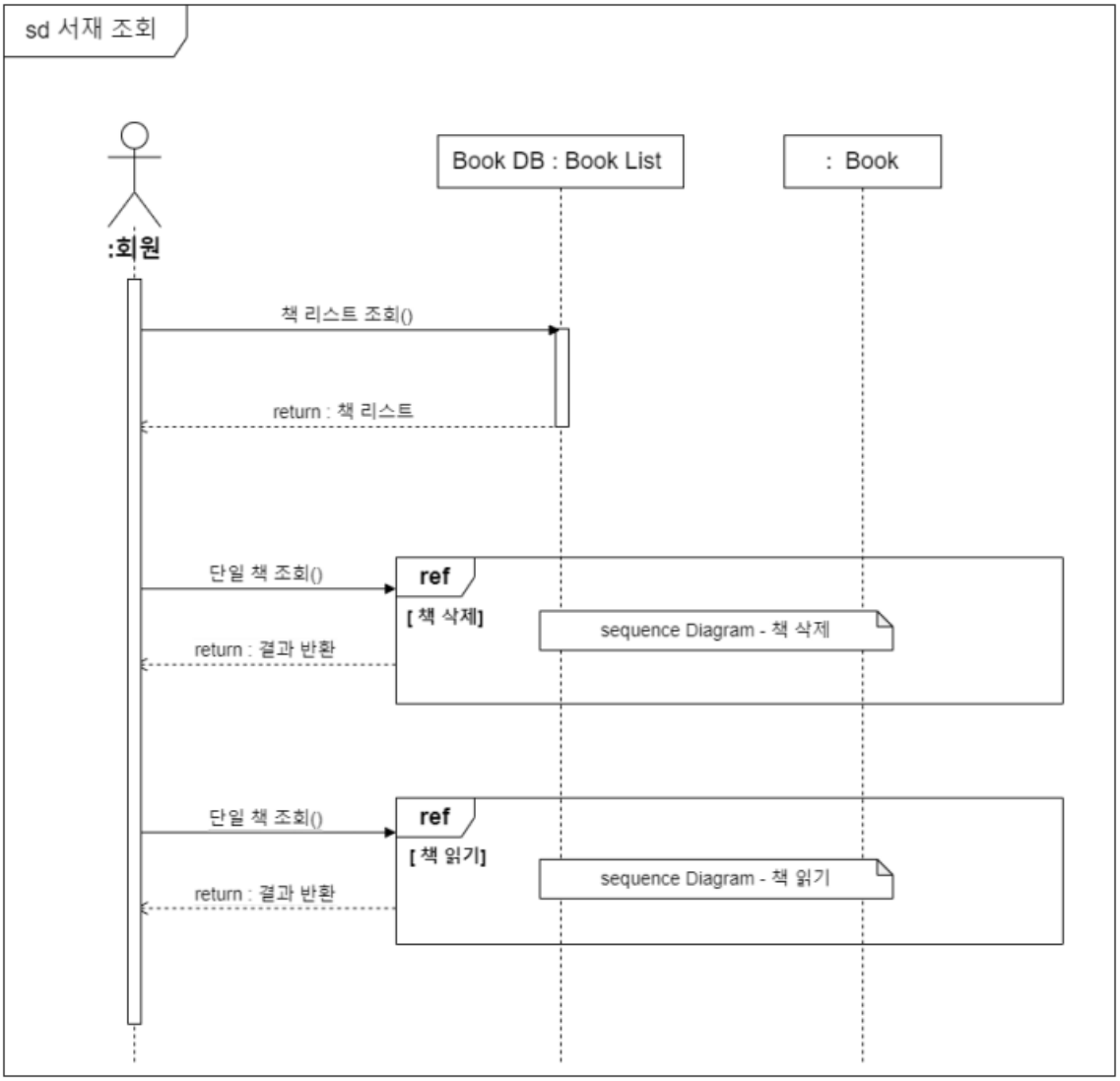
8.2 '회원관리(회원가입, 탈퇴, 회원 정보 수정)'에 대한 시퀀스 다이어그램



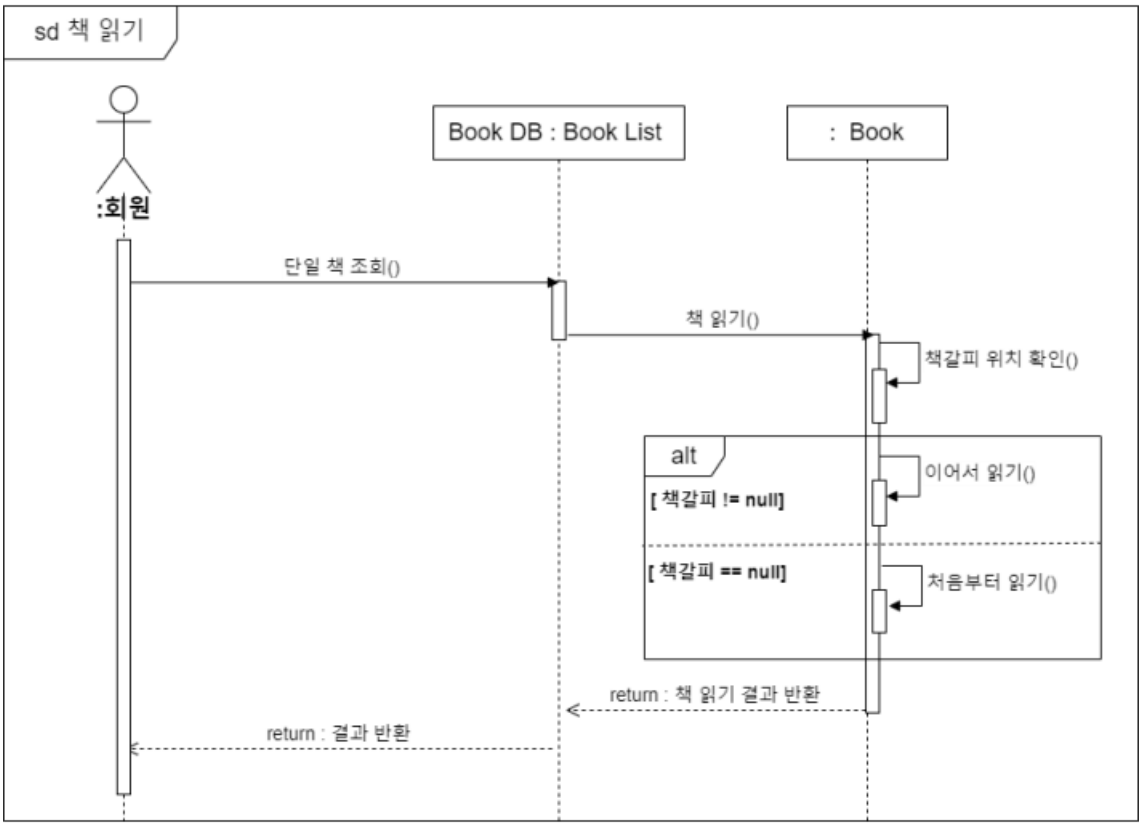
8.3 '회원 정보 수정'에 대한 시퀀스 다이어그램



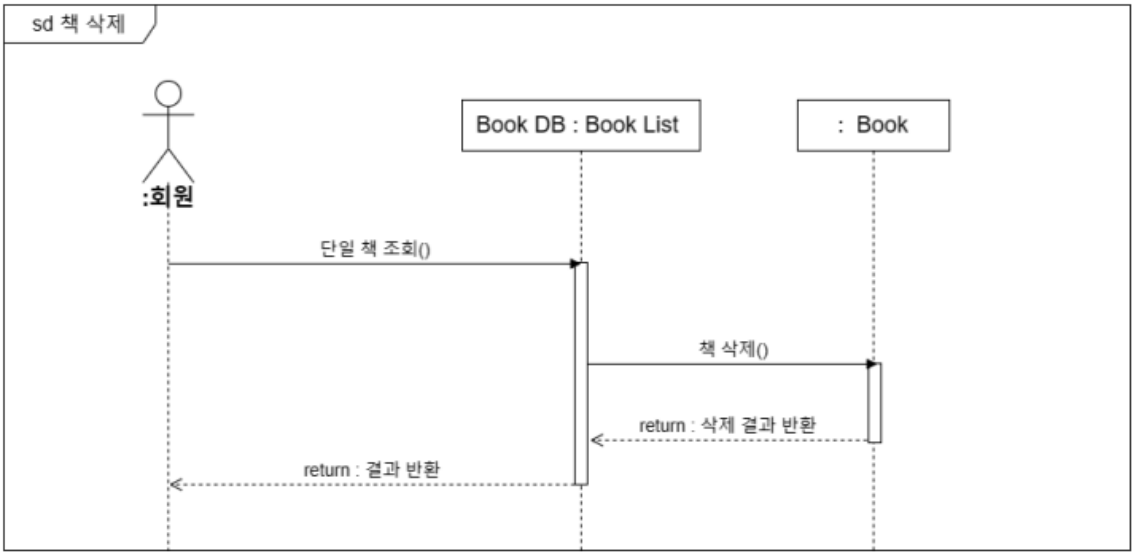
8.4 '서재조회'에 대한 시퀀스 다이어그램



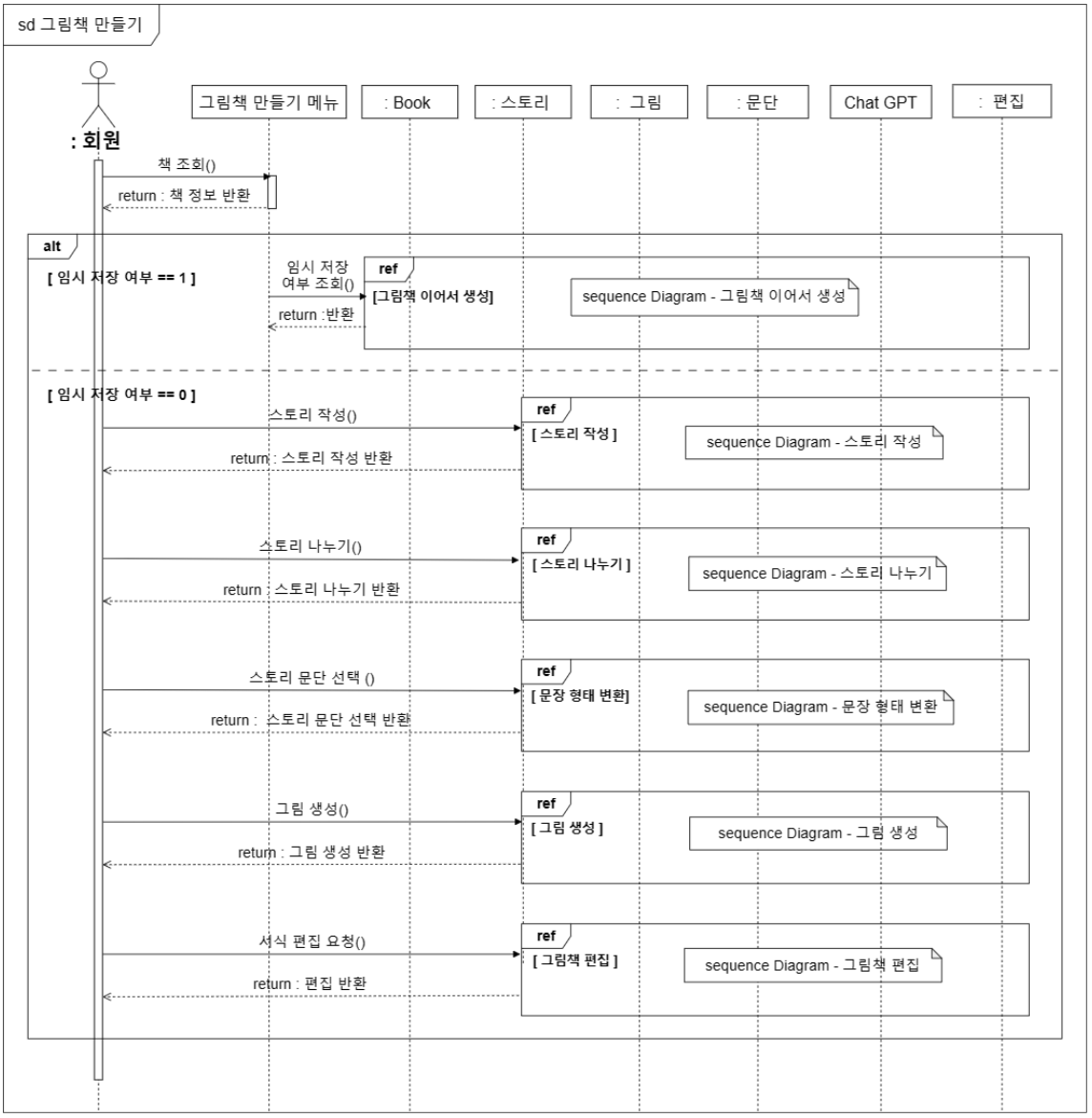
8.5 '책 읽기'에 대한 시퀀스 다이어그램



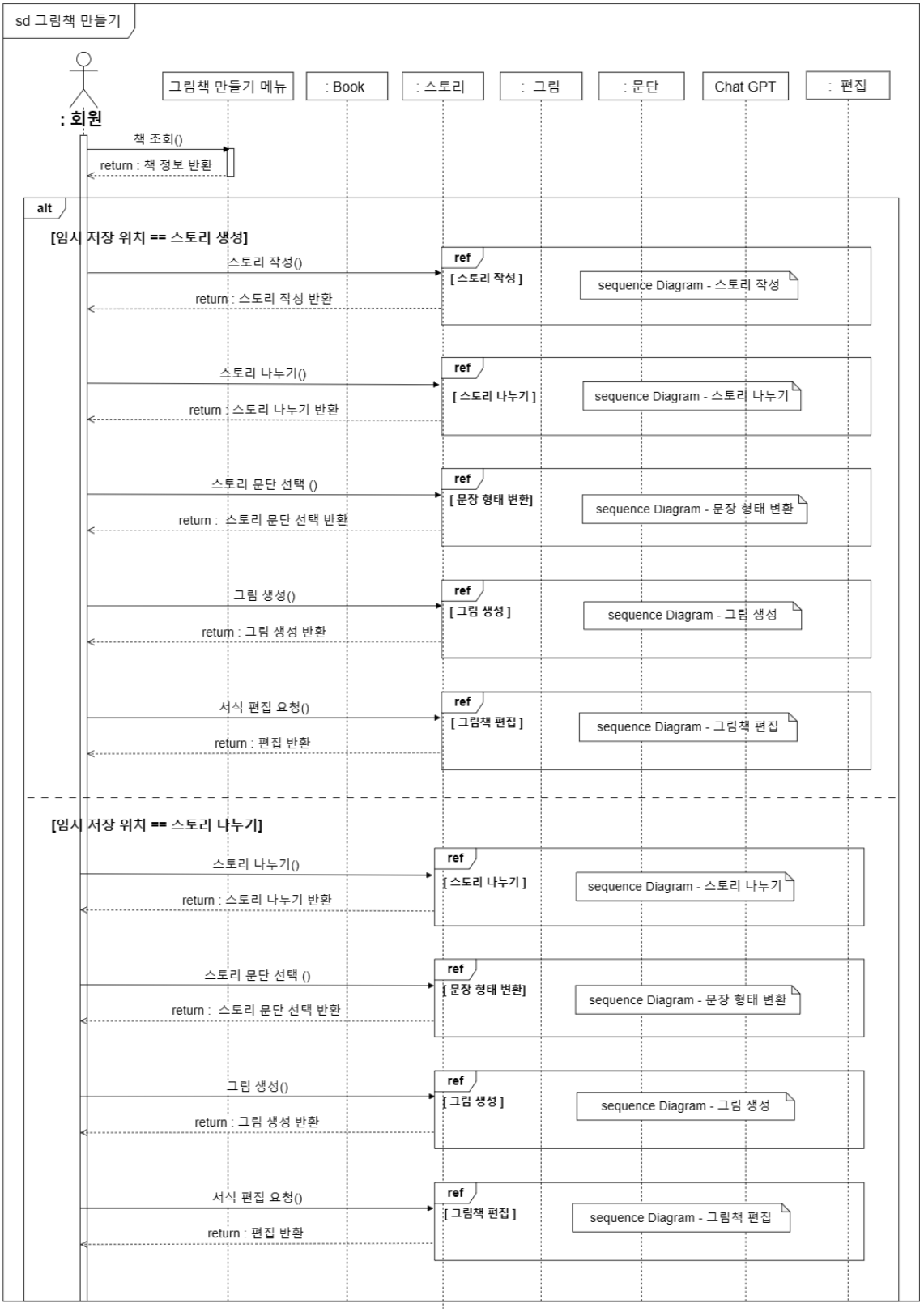
8.6 '책 삭제'에 대한 시퀀스 다이어그램



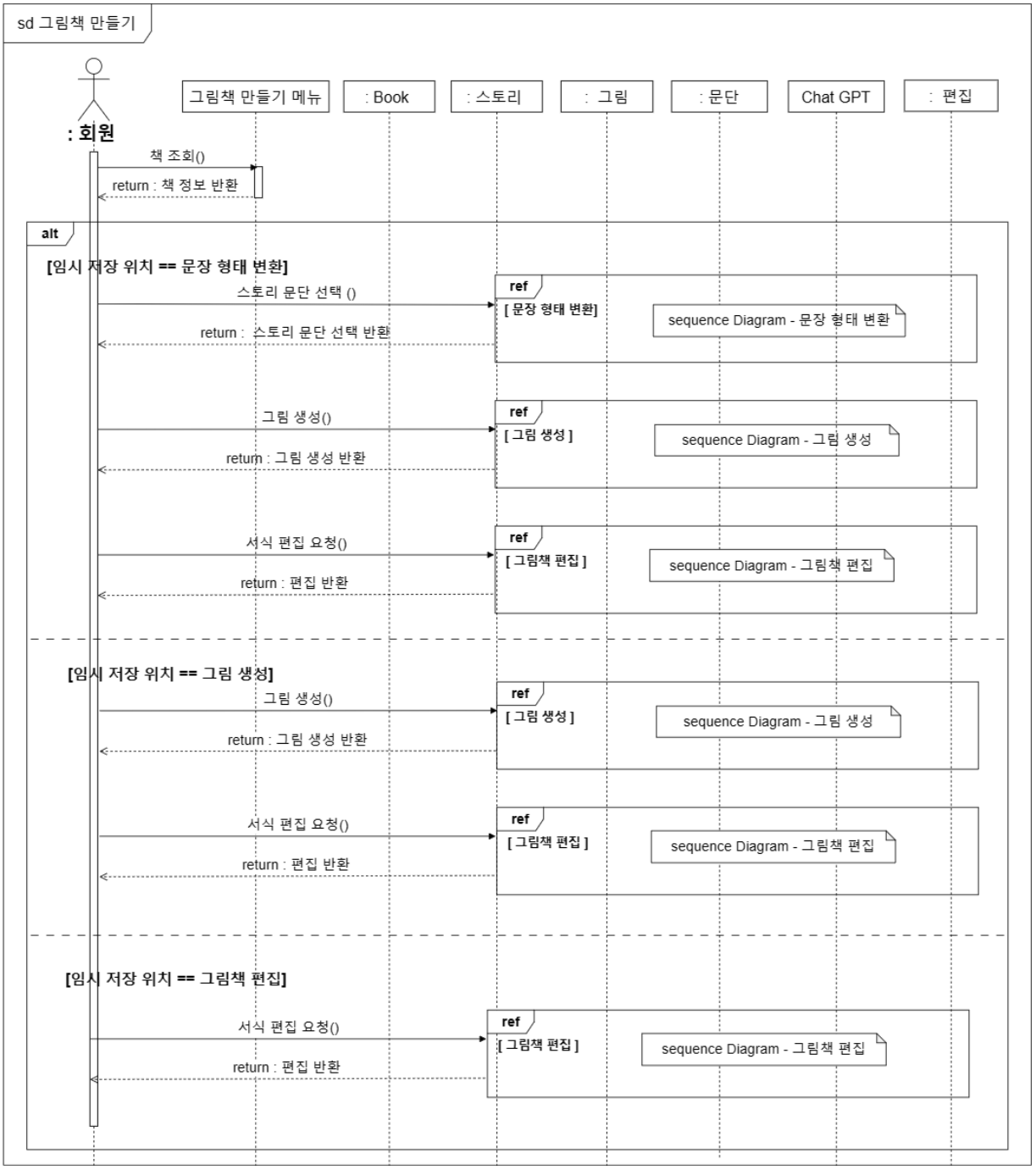
8.7 '그림책 만들기'에 대한 시퀀스 다이어그램



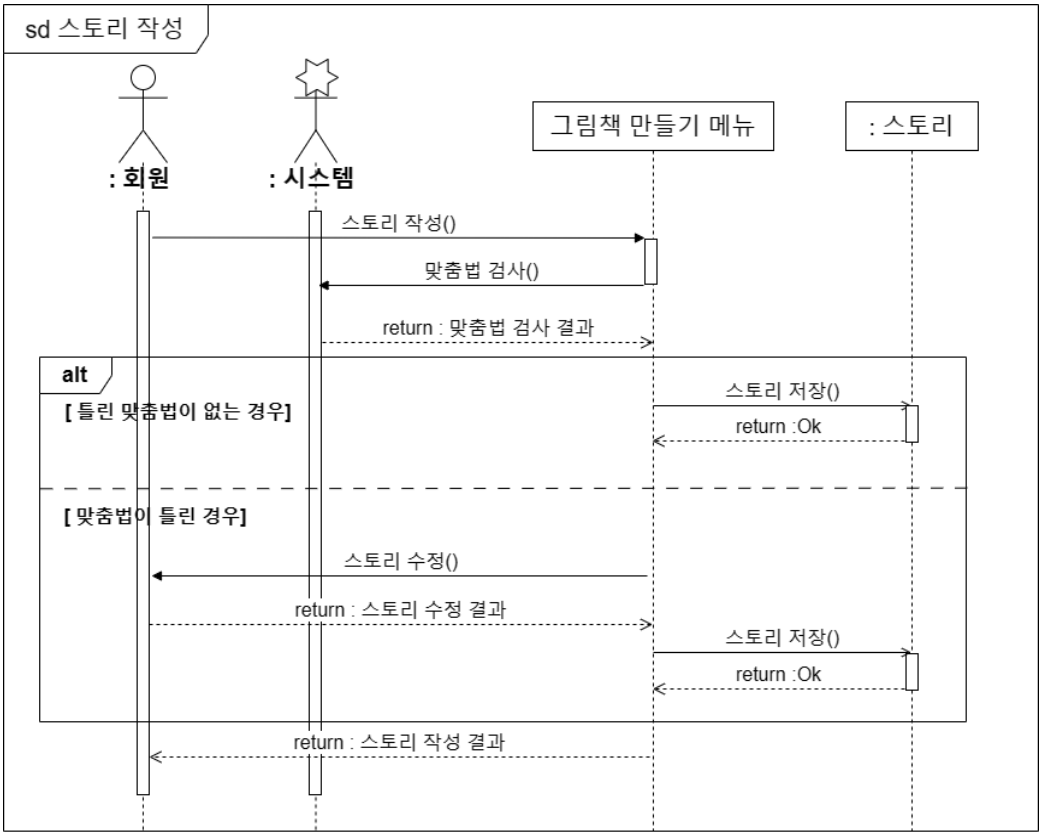
8.8 '그림책 이어서 생성'에 대한 시퀀스 다이어그램1



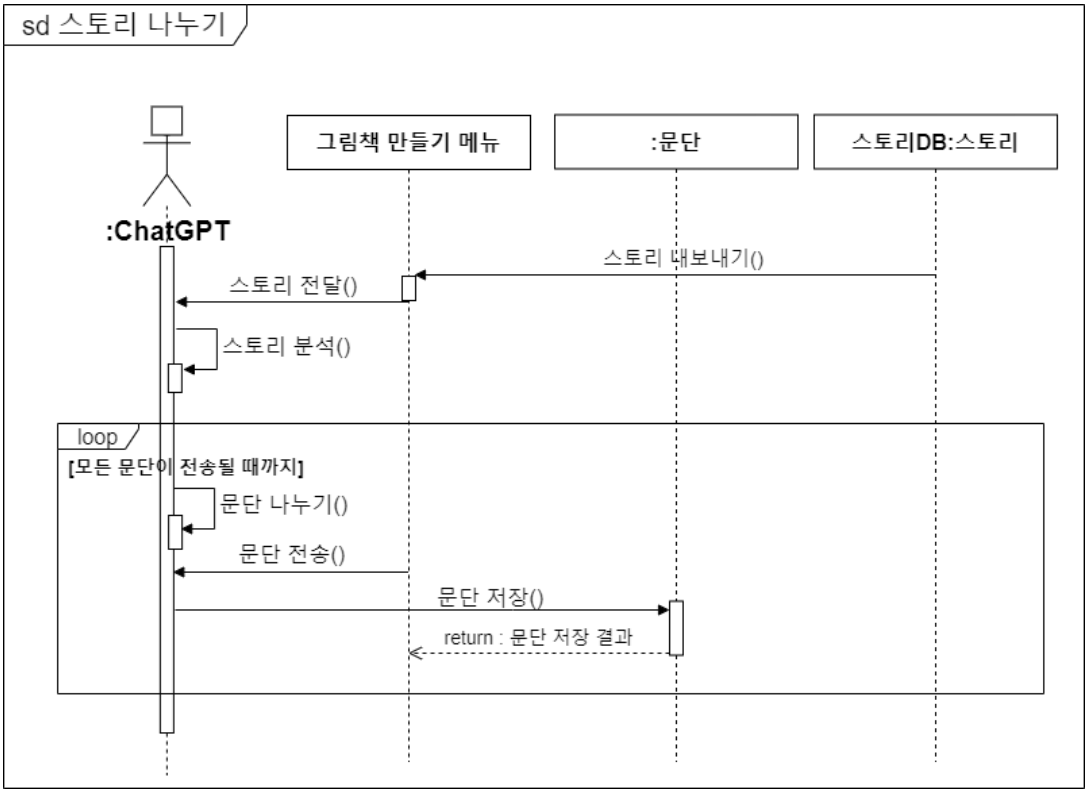
8.8.1 '그림책 이어서 생성'에 대한 시퀀스 다이어그램2



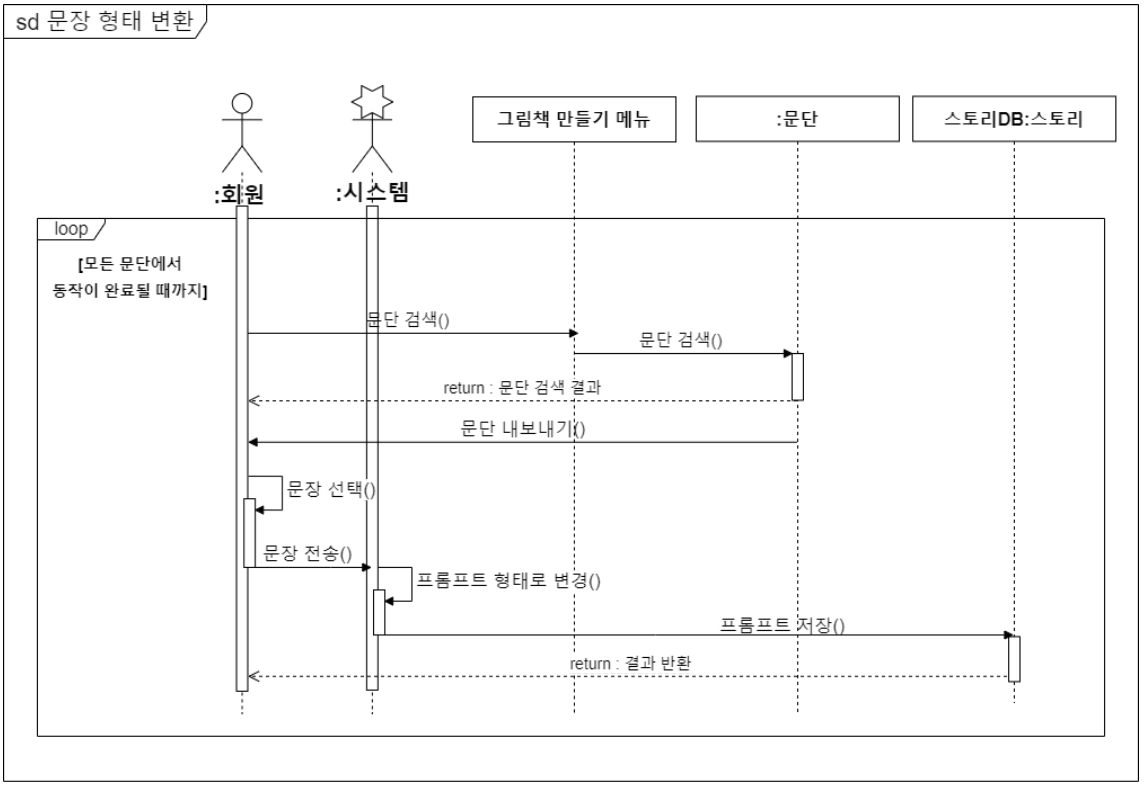
8.9 '스토리 작성'에 대한 시퀀스 다이어그램



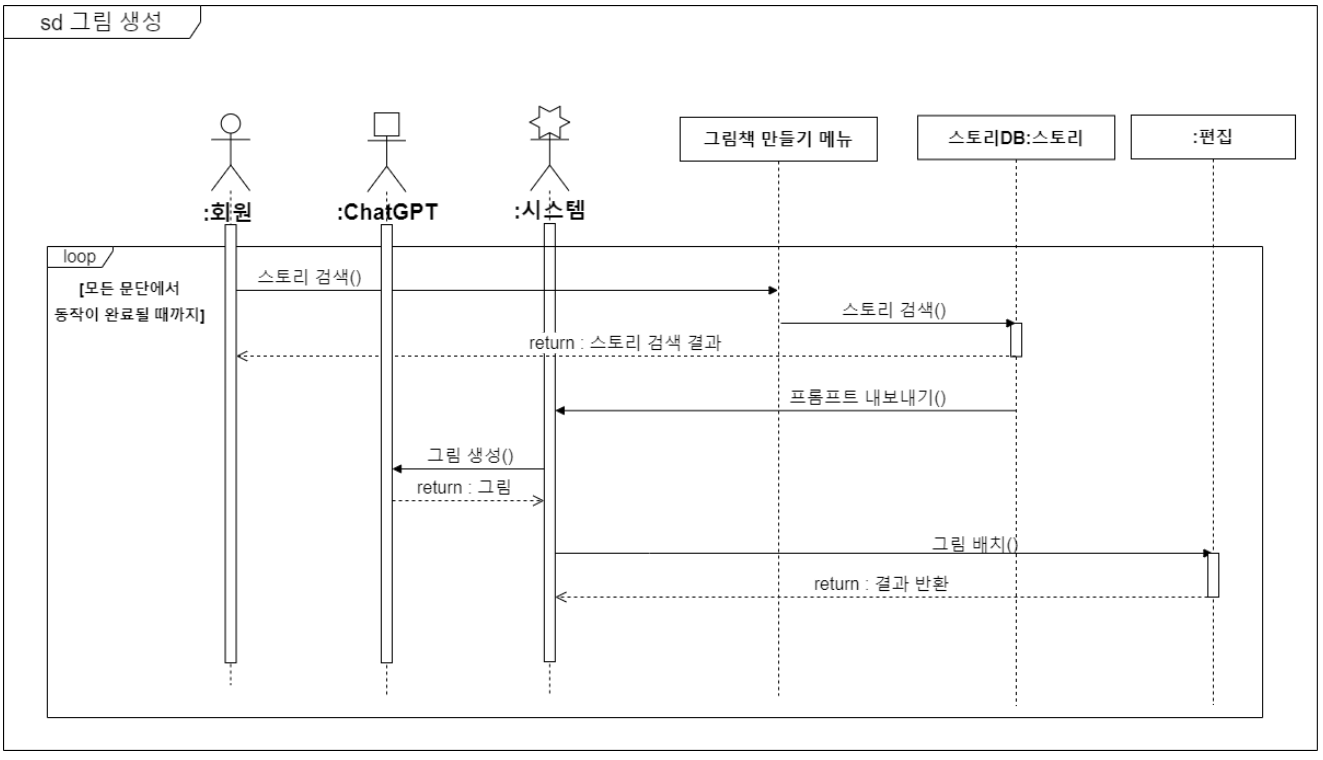
8.10 '스토리 나누기'에 대한 시퀀스 다이어그램



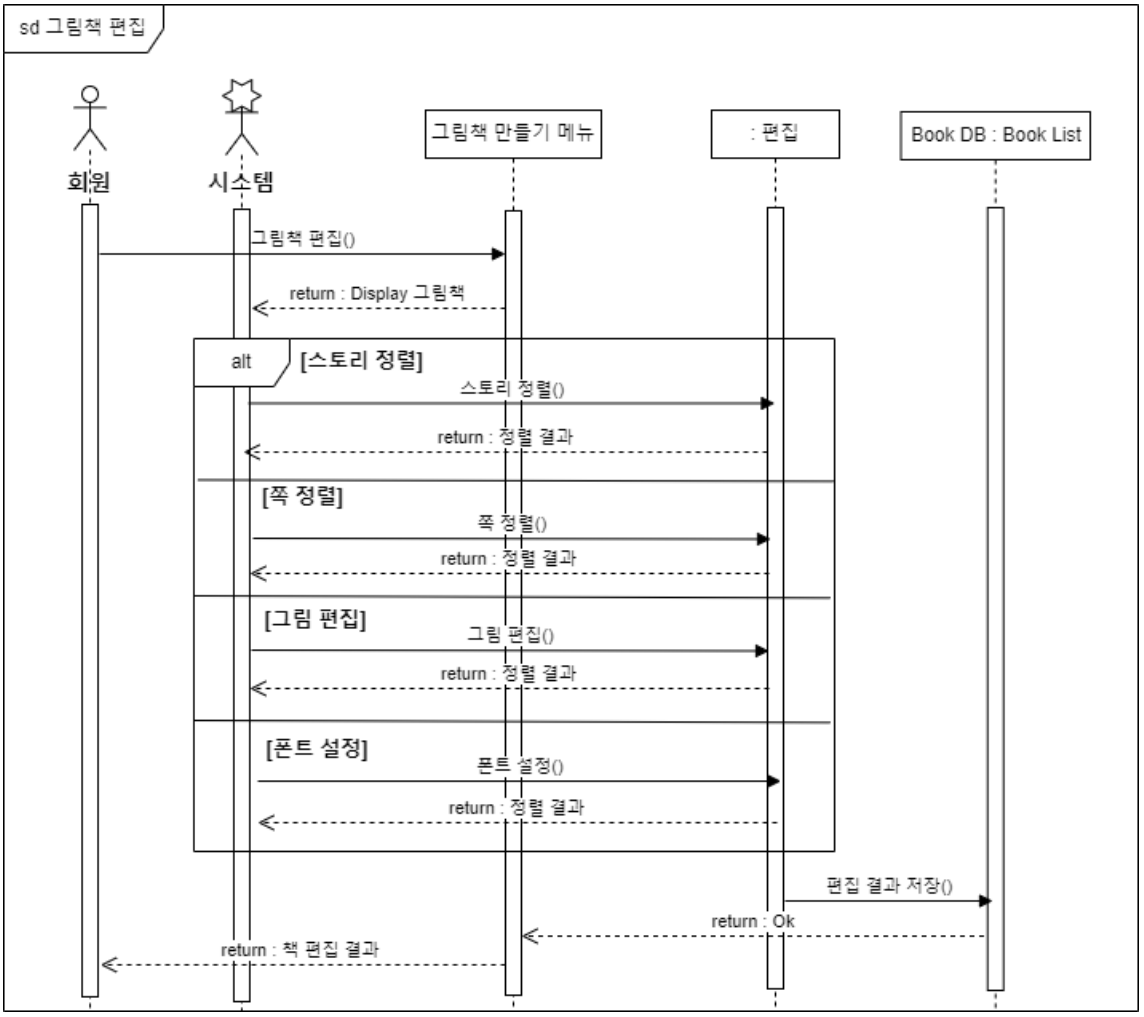
8.11 '문장 형태 변환'에 대한 시퀀스 다이어그램



8.12 '그림 생성'에 대한 시퀀스 다이어그램



8.13 '그림책 편집'에 대한 시퀀스 다이어그램



9. 클래스 메소드 알고리즘 명세

9.1 회원가입() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
메소드명: 회원가입

기능: 사용자의 회원가입을 처리하는 함수로, 입력된 사용자 아이디, 비밀번호, 닉네임을 이용하여 새로운 사용자를 생성하고 데이터베이스에 저장하고 이미 존재하는 사용자인 경우에는 회원가입을 실패로 처리함

파라미터: 사용자아이디 (user_id), 사용자비밀번호 (password), 사용자닉네임 (nickname)

반환값: 가입결과(result)
—----------------------------------------------------------------------------------------*

import java.util.HashMap;
import java.util.Map;

public class MembershipManager {



  // 가입 결과를 표현하는 열거형(Enum)

  public enum JoinResult {

    SUCCESS,

    DUPLICATE_USER,

    FAILURE

  }



  // 가입 정보를 담을 데이터베이스(Map)

  private Map<String, UserInfo> userDatabase;
```

```
// 사용자 정보를 담을 클래스
private static class UserInfo {
    String password;
    String nickname;

    public UserInfo(String password, String nickname) {
        this.password = password;
        this.nickname = nickname;
    }
}

// 생성자: 회원가입을 위한 초기화 작업 수행
public MembershipManager() {
    this.userDatabase = new HashMap<>();
}

/**
 * 회원가입을 처리하는 메소드
 *
 * @param user_id   사용자 아이디
 * @param password  사용자 비밀번호
```

```
 * @param nickname  사용자 닉네임
 * @return         가입 결과 (JoinResult)
 */
public JoinResult signUp(String user_id, String password, String nickname) {
    // 이미 존재하는 사용자인지 확인
    if (userDatabase.containsKey(user_id)) {
        return JoinResult.DUPLICATE_USER; // 이미 존재하는 사용자라면 가입 실패
    }

    // 새로운 사용자 생성 및 데이터베이스에 저장
    UserInfo newUser = new UserInfo(password, nickname);
    userDatabase.put(user_id, newUser);

    return JoinResult.SUCCESS; // 가입 성공
}

// 테스트용 메인 메소드
public static void main(String[] args) {
    MembershipManager membershipManager = new MembershipManager();

    // 가입 테스트
    JoinResult result1 = membershipManager.signUp("user1", "pass123", "nick1");
```

```
        System.out.println("가입 결과 1: " + result1); // 성공



        JoinResult result2 = membershipManager.signUp("user1", "pass456", "nick2");

        System.out.println("가입 결과 2: " + result2); // 중복 사용자로 실패



        JoinResult result3 = membershipManager.signUp("user2", "pass789", "nick3");

        System.out.println("가입 결과 3: " + result3); // 성공

    }

}
```

9.2 회원정보저장() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
메소드명: 회원정보저장

기능: 회원의 정보를 저장하는 함수로, 입력된 회원 닉네임, 아이디, 비밀번호를 이용하여 새로운 사용자를 생성하고 데이터베이스에 저장하고, 아이디가 이미 존재하는 경우에는 저장을 실패로 처리함

파라미터: 회원닉네임 (nickname), 아이디 (user_id), 비밀번호 (password)

반환값: 회원정보 저장 여부 (result)
----------------------------------------------------------------------------------------*

import java.util.HashMap;

import java.util.Map;
```

```
public class MembershipManager {


  // 저장 결과를 표현하는 열거형(Enum)
  public enum SaveResult {
    SUCCESS,
    DUPLICATE_USER,
    FAILURE
  }


  // 회원 정보를 담을 데이터베이스(Map)
  private Map<String, UserInfo> userDatabase;


  // 사용자 정보를 담을 클래스
  private static class UserInfo {
    String user_id;
    String password;
    String nickname;


    public UserInfo(String user_id, String password, String nickname) {
      this.user_id = user_id;
      this.password = password;
      this.nickname = nickname;
    }
  }
```

```
// 생성자: 회원 정보 저장을 위한 초기화 작업 수행
public MembershipManager() {
    this.userDatabase = new HashMap<>();
}

/**
 * 회원 정보를 저장하는 메소드
 *
 * @param nickname  회원 닉네임
 * @param user_id   회원 아이디
 * @param password  회원 비밀번호
 * @return          저장 결과 (SaveResult)
 */
    public SaveResult saveMemberInfo(String nickname, String user_id, String password) {
    // 이미 존재하는 사용자인지 확인
    if (userDatabase.containsKey(user_id)) {
            return SaveResult.DUPLICATE_USER; // 이미 존재하는 사용자라면 저장 실패
    }

     /      SaveResult result1 = membershipManager.saveMemberInfo("nick1", "user1", "pass123");
    System.out.println("저장 결과 1: " + result1); // 성공

            SaveResult result2 = membershipManager.saveMemberInfo("nick2", "user1", "pass456");
```

```
        System.out.println("저장 결과 2: " + result2); // 중복 사용자로 실패


                SaveResult result3 = membershipManager.saveMemberInfo("nick3",
"user2", "pass789");
        System.out.println("저장 결과 3: " + result3); // 성공
    }
}
```

9.3 사용자검색() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
메소드명: **사용자검색**
기능: 입력된 사용자 목록에서 특정 아이디를 검색하여 해당 사용자를 반환하는 함수
파라미터: 사용자목록(user_list), 검색아이디 (search_id)
반환값: 검색결과 (search_result)
----------------------------------------------------------------------------------------*
import java.util.List;


public class UserSearch {


    // 사용자 검색 결과를 표현하는 클래스
    public static class SearchResult {
        String user_id;
```

```
    String nickname;



    public SearchResult(String user_id, String nickname) {
        this.user_id = user_id;
        this.nickname = nickname;
    }



    @Override
    public String toString() {
        return "User ID: " + user_id + ", Nickname: " + nickname;
    }
}



/**
 * 사용자 검색 메소드
 *
 * @param user_list   사용자 목록
 * @param search_id   검색 아이디
 * @return            검색 결과 (SearchResult)
 */
    public static SearchResult searchUser(List<SearchResult> user_list, String
search_id) {
```

```
        for (SearchResult user : user_list) {
            if (user.user_id.equals(search_id)) {
                return user; // 검색 아이디와 일치하는 사용자를 찾았을 경우 반환
            }
        }
        return null; // 검색 결과가 없을 경우 null 반환
    }

    // 테스트용 메인 메소드
    public static void main(String[] args) {
        // 사용자 목록 생성
        List<SearchResult> userList = List.of(
                new SearchResult("user1", "nick1"),
                new SearchResult("user2", "nick2"),
                new SearchResult("user3", "nick3")
        );

        // 사용자 검색 테스트
        String searchId1 = "user2";
        SearchResult result1 = searchUser(userList, searchId1);
            System.out.println("검색 결과 1: " + (result1 != null ? result1.toString() :
"사용자를 찾을 수 없습니다."));
```

```
        String searchId2 = "user4";
        SearchResult result2 = searchUser(userList, searchId2);
            System.out.println("검색 결과 2: " + (result2 != null ? result2.toString() :
"사용자를 찾을 수 없습니다."));
    }
}
```

9.4 스토리 생성 및 맞춤법 검사() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
메소드명: 스토리 생성 및 맞춤법 검사
기능: 주어진 스토리 내용을 작성하고 맞춤법을 검사하여 결과를 반환하는 함수
파라미터: 스토리내용 (story_content)
반환값: 맞춤법검사결과 (spelling_result)
----------------------------------------------------------------------------------------*
import org.languagetool.JLanguageTool;
import org.languagetool.Language;
import org.languagetool.language.English;

import java.util.List;

public class StoryGenerator {
```

```
// 맞춤법 검사 결과를 표현하는 클래스
public static class SpellingResult {
    boolean isCorrect;
    List<String> suggestions;

    public SpellingResult(boolean isCorrect, List<String> suggestions) {
        this.isCorrect = isCorrect;
        this.suggestions = suggestions;
    }

    @Override
    public String toString() {
        if (isCorrect) {
            return "맞춤법 검사 결과: 문제 없음";
        } else {
            return "맞춤법 검사 결과: 틀린 부분이 있습니다. 제안 사항: " + suggestions;
        }
    }
}

/**
 * 스토리 생성 및 맞춤법 검사 메소드
 *
```

```
 * @param story_content 스토리 내용
 * @return              맞춤법 검사 결과 (SpellingResult)
 */
public static SpellingResult generateAndCheckStory(String story_content) {
    // 스토리 생성 로직은 생략하고, 맞춤법 검사만 수행
    Language language = new English(); // 검사할 언어 설정 (예: 영어)
    JLanguageTool languageTool = new JLanguageTool(language);

    try {
        // 맞춤법 검사 수행
        List<String> errors = languageTool.check(story_content);

        // 검사 결과에 따라 SpellingResult 객체 생성
        if (errors.isEmpty()) {
            return new SpellingResult(true, null); // 맞춤법 오류가 없는 경우
        } else {
            return new SpellingResult(false, errors); // 맞춤법 오류가 있는 경우
        }
    } catch (Exception e) {
        e.printStackTrace();
        return new SpellingResult(false, null); // 검사 중 오류 발생 시 처리
    }
}
```

```
    // 테스트용 메인 메소드
    public static void main(String[] args) {
        // 테스트용 스토리 내용
         String testStory = "Once upon a time, there was a boy who lived in a beautful village.";

        // 스토리 생성 및 맞춤법 검사 테스트
        SpellingResult result = generateAndCheckStory(testStory);
        System.out.println(result.toString());
    }
}
```

9.5 스토리 수정() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
메소드명: modifyStory
기능: 주어진 현재 스토리를 수정하고 수정된 스토리를 반환하는 함수
파라미터: 현재스토리 (current_story)
반환값: 수정된스토리 (modified_story)
----------------------------------------------------------------------------------------*
public class Main {
    public static void main(String[] args) {
        Story currentStory = new Story(); // 현재 스토리
```

```
    String newText = "새로운 텍스트"; // 수정할 텍스트
    int storyNumber = 1; // 수정할 스토리 번호

    Story modifiedStory = modifyStory(currentStory, newText, storyNumber);

    // 수정된 스토리 출력
    if (modifiedStory != null) {
        System.out.println("수정된 스토리: " + modifiedStory.getText());
    } else {
        System.out.println("해당 스토리가 수정되지 않았습니다.");
    }
}

// 스토리 수정 함수
public static Story modifyStory(Story currentStory, String newText, int storyNumber) {
    Story currentPage = getCurrentStory(storyNumber); // 현재 페이지

    // 새로운 텍스트가 주어진 경우
    if (newText != null && !newText.isEmpty()) {
        currentPage.setText(newText); // 텍스트 수정
        return currentPage;
    } else {
        return currentStory;
```

```
        }
    }

    // 현재 스토리 가져오기 함수
    public static Story getCurrentStory(int storyNumber) {
        // 현재 스토리를 가져오는 로직
        // 예시로, 빈 스토리 객체 반환
        return new Story();
    }
}

class Story {
    private String text;

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }
}
```

9.6 문장선택() 알고리즘 명세

*----------

메소드명: selectSentence

기능: 입력된 문단 목록에서 사용자로부터 특정 문장의 번호를 입력받아 해당 문장을 반환하는 함수

파라미터: 나눠진 문단 (divided_paragraphs), 선택된 문장 (selected_sentence)

반환값: 선택된 문장 (selected_sentence)

----------*

```
import java.util.Scanner;



public class Main {
    public static void main(String[] args) {
        String[] paragraph = {"첫번째 문장", "두번째 문장", "세번째 문장"}; // 예시 문단
        String selectedSentence = selectSentence(paragraph);
        System.out.println("선택된 문장: " + selectedSentence); // 선택된 문장 출력
    }



    // 사용자에게 문장을 선택받는 함수
    public static String selectSentence(String[] paragraph) {
        Scanner scanner = new Scanner(System.in); // 입력을 위한 스캐너
        String selectedSentence = ""; // 선택된 문장을 저장할 변수



        // 각 문단을 출력
```

```
        for (int i = 0; i < paragraph.length; i++) {
            System.out.println("(" + (i + 1) + ") " + paragraph[i]);
        }

        System.out.println("선택할 문장의 번호를 입력하세요: "); // 사용자에게 번호 입력 요청
        int selectedNumber = scanner.nextInt(); // 사용자로부터 번호 입력 받음

        // 입력받은 번호가 범위 안에 있는지 확인
        if (selectedNumber >= 1 && selectedNumber <= paragraph.length) {
            selectedSentence = paragraph[selectedNumber - 1]; // 범위 안에 있으면 해당 문장 선택
        } else {
            System.out.println("올바른 번호를 입력하세요."); // 범위 밖이면 오류 메시지 출력
        }

        return selectedSentence; // 선택된 문장 반환
    }
}
```

9.7 회원정보수정() 알고리즘 명세

*----------

메소드명: updateUser

기능: 주어진 사용자 아이디와 새로운 정보를 이용하여 회원 정보를 수정하고, 수정된 회원 정보를 반환하는 함수

파라미터: 사용자 아이디 (userId), 새로운 정보 (newInfo)

반환값: 수정된 회원 정보 (modified_user_info)

----------*

```
public User updateUser(String userId, UserInfo newInfo) {
    // 입력받은 아이디를 이용하여 회원 정보를 조회
    User user = userService.findUserById(userId);

    // 회원 정보가 존재하는 경우
    if (user != null) {
        // 회원 정보 수정 처리
        user = userService.updateUser(user, newInfo);
        // 수정된 회원 정보를 반환
        return user;
    } else {
        // 회원 정보가 없는 경우, null 반환
        return null;
    }
}
```

9.8 회원삭제() 알고리즘 명세

*----------

메소드명: deleteUser

기능: 주어진 사용자 아이디를 이용하여 회원 정보를 삭제하고, 삭제 결과를 반환하는 함수

파라미터: 사용자 아이디 (user_id)

반환값: 삭제 결과 (deletion_result)

----------*

```
public boolean deleteUser(String userId) {
    // 입력받은 아이디를 이용하여 회원 정보를 조회
    User user = userService.findUserById(userId);

    // 회원 정보가 존재하는 경우
    if (user != null) {
        // 회원 삭제 처리
        userService.deleteUser(user);
        // 삭제 결과를 반환 (성공)
        return true;
    } else {
        // 회원 정보가 없는 경우, 삭제 결과를 반환 (실패)
        return false;
    }
}
```

9.9 회원탈퇴() 알고리즘 명세

*----------

메소드명: withdrawMembership

기능: 주어진 사용자 아이디와 비밀번호를 이용하여 회원 정보를 탈퇴하고, 탈퇴 결과를 반환하는 함수

파라미터: 사용자 아이디 (user_id), 사용자 비밀번호 (user_password)

반환값: 탈퇴 결과 (withdrawal_result)

----------*

```
public boolean withdrawMembership(String userId, String userPassword) {
   // 입력받은 아이디와 비밀번호를 이용하여 회원 정보를 조회
   User user = userService.findUser(userId, userPassword);

   // 회원 정보가 존재하는 경우
   if (user != null) {
      // 회원 탈퇴 처리
      userService.withdrawUser(user);
      // 탈퇴 결과를 반환 (성공)
      return true;
   } else {
      // 회원 정보가 없는 경우, 탈퇴 결과를 반환 (실패)
      return false;
   }
```

```
}
```

9.10 임시저장여부조회() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------

메소드명: checkTemporaryStorageStatus

기능: 주어진 책 정보를 이용하여 해당 책이 임시 저장되어 있는지 여부를 조회하고, 그 결과를 반환하는 함수

파라미터: 책 정보 (book_info)

반환값: 임시 저장 여부 (temporary_storage_status)

-----------------------------------------------------------------------------------------*

public boolean checkTemporaryStorageStatus(BookInfo bookInfo) {

   // 임시 저장 여부를 가져옴

   boolean temporaryStorageStatus = bookInfo.getTemporaryStorageStatus();



   // 임시 저장 여부를 반환

   return temporaryStorageStatus;

}
```

9.11 임시저장 위치 확인() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------

메소드명: checkTemporaryStorageLocation

기능: 주어진 책 정보를 이용하여 해당 책이 임시 저장된 위치를 확인하고, 그 결과를 반환하는 함수

파라미터: 책 정보 (book_info)
```

반환값: 임시 저장 위치 (temporary_storage_location)

---*

```
public String checkTemporaryStorageLocation(BookInfo bookInfo) {
    // 임시 저장 위치를 가져옴
    String temporaryStorageLocation = bookInfo.getTemporaryStorageLocation();

    // 임시 저장 위치를 반환
    return temporaryStorageLocation;
}
```

9.12 책갈피 생성() 알고리즘 명세

*---

메소드명: createBookmark

기능: 주어진 책 정보와 책갈피 위치를 이용하여 책갈피를 생성하는 함수

파라미터: 책 정보 (book_info), 책갈피 위치 (bookmark_location)

반환값: 없음

---*

```
public void createBookmark(BookInfo bookInfo, int bookmarkLocation) {
    // 책갈피를 생성
```

```
    bookInfo.setBookmarkPage(bookmarkLocation);



    // 책갈피 생성 완료 메시지를 출력

    System.out.println("Bookmark created at page " + bookmarkLocation + ".");

}
```

9.13 책읽기() 알고리즘 명세

*----------

메소드명: readBook

기능: 주어진 책 정보를 이용하여 사용자에게 다양한 동작을 선택하도록 하여 책을 읽는 함수이다.

파라미터: 책 정보 (book_info)

반환값: 없음

----------*

```
public void readBook(BookInfo bookInfo) {

    // 사용자에게 동작을 선택하도록 메뉴를 표시

    System.out.println("Select an action:");

    System.out.println("1. Start reading from the beginning");

    System.out.println("2. Continue reading from the bookmark");



    // 사용자의 선택을 입력받음

    Scanner scanner = new Scanner(System.in);

    int action = scanner.nextInt();
```

```
    switch(action) {
        case 1:
            // 사용자가 '처음부터 읽기'를 선택한 경우, 해당 함수를 호출
            readFromBeginning(bookInfo);
            break;
        case 2:
            // 사용자가 '이어서 읽기'를 선택한 경우, 해당 함수를 호출
            continueReading(bookInfo);
            break;
        default:
            // 사용자가 잘못된 선택을 한 경우, 에러 메시지를 출력
            System.out.println("Invalid action selected.");
            break;
    }
}
```

9.14 이어서읽기**()** 알고리즘 명세

*----------

메소드명: continueReading

기능: 주어진 책 정보를 이용하여 책갈피의 위치에서 이어서 페이지를 읽는 함수

파라미터: 책 정보 (book_info)

반환값: 없음

---*

```
public void continueReading(BookInfo bookInfo) {
   // 책갈피의 위치를 가져옴
   int bookmarkPage = bookInfo.getBookmarkPage();

   // 책갈피의 위치가 존재하는지 확인
   if (bookmarkPage != 0) {
      // 책갈피의 위치에서 이어서 읽기 시작하는 메시지를 출력
       System.out.println("Continuing to read the book from page " + bookmarkPage + ".");
   } else {
      // 책갈피의 위치가 없으면, 처음부터 읽기 시작하는 메시지를 출력
        System.out.println("No bookmark found. Starting to read the book from the beginning.");
   }
}
```

9.15 처음부터읽기() 알고리즘 명세

*----------

메소드명: readFromBeginning

기능: 주어진 책 정보를 이용하여 처음부터 페이지를 읽는 함수로, 책갈피 내역이 없으면

새로 시작함

파라미터: 책 정보 (book_info)

반환값: 없음

---*

```
public void readFromBeginning(BookInfo bookInfo) {
    // 책갈피 내역이 있는지 확인
    if (bookmarkHistory.isEmpty()) {
        // 책갈피 내역이 없으면, 메시지를 출력
        System.out.println("There is no saved bookmark. Starting from the beginning.");
    }

    // 책을 처음부터 읽기 시작하는 메시지를 출력
    System.out.println("Starting to read the book from the beginning.");
    // 현재 페이지를 1로 설정
    int currentPage = 1;
}
```

9.16 책삭제() 알고리즘 명세

*----------

메소드명: deleteBook

기능: 주어진 책 아이디를 이용하여 책을 삭제하는 함수

파라미터: 책 아이디 (book_id)

반환값: 없음

----------*

```
public void deleteBook(String bookId) {
    // 책이 데이터베이스에 존재하는지 확인
    boolean bookExists = checkBookExists(bookId);

    if (bookExists) {
        // 책이 존재하면, 책을 삭제
        performBookDeletion(bookId);
        // 삭제 성공 메시지를 출력
        System.out.println("The book has been successfully deleted.");
    } else {
        // 책이 존재하지 않으면, 에러 메시지를 출력
        System.out.println("The book does not exist.");
    }
}
```

9.17 책조회() 알고리즘 명세

*----------

메소드명: retrieveBook

기능: 주어진 책 아이디를 이용하여 단일 책을 조회하는 함수로, 책이 존재하면 해당 책의 정보를 반환하고, 책이 존재하지 않으면 null을 반환함

파라미터: 책 아이디 (book_id)

반환값: 책 정보 (book_info)

----------*

```
public BookInfo retrieveBook(String bookId) {
   // 책이 데이터베이스에 존재하는지 확인
   boolean bookExists = checkBookExists(bookId);

   // 책 정보를 저장할 객체를 선언
   BookInfo bookInfo;

   if (bookExists) {
      // 책이 존재하면, 책 정보를 조회
      bookInfo = getBookInfo(bookId);
      // 조회된 책 정보를 출력
      System.out.println("Book retrieval successful: " + bookInfo);
   } else {
```

```
        // 책이 존재하지 않으면, 에러 메시지를 출력
        System.out.println("The book does not exist.");
        // bookInfo를 null로 설정
        bookInfo = null;
    }

    // 책 정보를 반환
    return bookInfo;
}
```

9.18 책저장() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------
메소드명: saveBook
기능: 주어진 책 정보를 이용하여 책을 저장하는 함수이다. 책 아이디를 생성하고, 해당 아이디를 책 정보에 설정한 후 데이터베이스 또는 저장소에 책 정보를 저장함
파라미터: book_info
반환값: 없음
-----------------------------------------------------------------------------------------*
public void saveBook(BookInfo book_info) {
    // 책 아이디를 생성
```

```
    // 이 부분은 실제 책 아이디 생성 로직에 따라 구현됩니다.
    // 아래는 예시로 작성한 의사코드입니다.
    String book_id = generateBookId();

    // 생성된 아이디를 책 정보에 설정
    book_info.setBookId(book_id);

    // 설정된 책 정보를 데이터베이스 또는 저장소에 저장
    // 이 부분은 실제 데이터베이스 또는 저장소와의 연동 방식에 따라 구현됩니다.
    // 아래는 예시로 작성한 의사코드입니다.
    database.saveBookInfo(book_info);
}
```

9.19 문단나누기() 알고리즘 명세

*----------

메소드명: divideParagraphs

기능: 주어진 스토리를 문장 단위로 나누어 문단으로 구분하는 함수이다. 문장이 마침표, 느낌표, 물음표로 끝나면 그 문장까지를 하나의 문단으로 간주함

파라미터: generated_story

반환값: 나눠진 문단 (divided_paragraphs)

----------*

```
public List<String> divideParagraphs(String generated_story) {
    // 결과를 저장할 리스트를 생성
    List<String> divided_paragraphs = new ArrayList<>();
```

```
    // 문장 단위로 나누는 정규식을 정의 (. ? ! 뒤에 공백이 있는 부분으로 나눔)
    String regex = "(?<=[.!?])\\s+";

    // 정규식을 기반으로 스토리를 문장 단위로 나눔
    String[] sentences = generated_story.split(regex);

    // 각 문장을 리스트에 추가
    for (String sentence : sentences) {
        divided_paragraphs.add(sentence);
    }

    // 나눠진 문단을 반환
    return divided_paragraphs;
}
```

9.20 그림_생성() 알고리즘 명세

*----------

메소드명: createPicture

기능: 주어진 프롬프트를 기반으로 그림을 생성하는 함수로, 구체적인 로직이나 구현은 슈도코드에 제공되지 않음

파라미터: prompt

반환값: 생성된 그림

—----------*

```
public Picture createPicture(String prompt) {
    // Picture는 그림을 나타내는 객체입니다.

    // 그림 생성 로직
    // 이 부분은 실제 그림 생성 알고리즘에 따라 구현됩니다.
    // 아래는 예시로 작성한 의사코드입니다.

    // 새 그림 객체를 생성
    Picture picture = new Picture();

    // 프롬프트를 기반으로 그림 생성 로직을 수행
    // 이 부분은 실제 그림 생성 알고리즘에 따라 구현됩니다.
    // 아래는 그림 생성 로직을 표현하기 위한 의사코드입니다.
    picture.drawBasedOnPrompt(prompt);

    // 생성된 그림을 반환
    return picture;
}
```

9.21 책_글꼴_설정() 알고리즘 명세

*----------

메소드명: setBookFont

기능: 주어진 책 정보에 선택한 글꼴을 설정하고, 이를 데이터베이스 또는 저장소에 업데이트하는 함수

파라미터: bookInfo, selectedFont

반환값: 없음 (출력이 없는 함수)

—----------*

```
public void setBookFont(BookInfo bookInfo, String selectedFont) {
    // 책 정보 객체에 선택한 글꼴을 설정
    bookInfo.setFont(selectedFont);

    // 설정한 글꼴을 데이터베이스 또는 저장소에 업데이트
    // 이 부분은 실제 데이터베이스 또는 저장소와의 연동 방식에 따라 구현됩니다.
    // 아래는 예시로 작성한 의사코드입니다.
    database.updateBookInfo(bookInfo);
}
```

9.22 텍스트_정렬() 알고리즘 명세

*----------

메소드명: textAlignment

기능: 주어진 텍스트를 지정된 정렬 방식에 따라 정렬하는 함수로, 정렬 방식에 따라 가운데

```
정렬, 왼쪽 정렬, 오른쪽 정렬을 수행함

파라미터:  text,  alignment

반환값: 정렬된 텍스트

—----------------------------------------------------------------------------------------*

public String textAlignment(String text, String alignment) {



   // Screen width assumed for alignment

   int screenWidth = 80;



   // Calculate the length of the given text

   int textLength = text.length();



   // Initialize the number of spaces to add in front

   int numSpaces = 0;



   // Use StringBuilder to add space characters

   StringBuilder sb = new StringBuilder();



   switch(alignment) {

      case "center":

         // Calculate the number of spaces to add in front for center alignment

         numSpaces = (screenWidth - textLength) / 2;

         break;
```

```
        case "left":
            // No spaces needed for left alignment
            numSpaces = 0;
            break;
        case "right":
            // Calculate the number of spaces to add in front for right alignment
            numSpaces = screenWidth - textLength;
            break;
        default:
            // If an unknown alignment is entered, return the original text
            return text;
    }
    for(int i = 0; i < numSpaces; i++){
        sb.append(" ");
    }
    // Append the given text to the StringBuilder
    sb.append(text);


    // Return the aligned text
    return sb.toString();
}
```

9.23 가운데_정렬**()** 알고리즘 명세

*----------

메소드명: centerAlign

기능: 주어진 텍스트를 가운데 정렬하는 함수로, 가운데 정렬된 텍스트를 반환함

파라미터: text

반환값: 가운데 정렬된 텍스트

—----------*

```
public String centerAlign(String text) {
    // Screen width assumed for center alignment
    int screenWidth = 80;


    // Calculate the length of the given text
    int textLength = text.length();


    // Calculate the number of spaces to add in front for center alignment
    // (screenWidth - textLength) / 2
    int numSpaces = (screenWidth - textLength) / 2;


    // Use StringBuilder to add space characters in front
    StringBuilder sb = new StringBuilder();


    for(int i = 0; i < numSpaces; i++){
        sb.append(" ");
```

```
    }


    // Append the given text to the StringBuilder
    sb.append(text);


    // Return the center-aligned text
    return sb.toString();
}
```

9.24 왼쪽_정렬() 알고리즘 명세

*----------

메소드명: 왼쪽_정렬

기능: 주어진 텍스트를 왼쪽 정렬하는 함수로, 왼쪽 정렬된 텍스트를 반환함

파라미터: text

반환값: 왼쪽 정렬된 텍스트

—----------*

```
public String alignLeft(String text) {
    // 함수 정의: 주어진 텍스트를 왼쪽 정렬함


    int screenWidth = 80; // 화면 너비를 예시로 80으로 설정
     int paddingSize = screenWidth - text.length(); // 패딩 사이즈를 계산 (화면 너비에서 텍스트 길이를 뺌)


    if (paddingSize < 0) { // 패딩 사이즈가 음수일 경우 (텍스트가 화면 너비보다 긴 경우)
```

```
        return text; // 텍스트를 그대로 반환
    }

        String padding = String.format("%" + paddingSize + "s", ""); // 패딩 문자열 생성 (공백으로 채움)
    String result = text + padding; // 텍스트와 패딩 문자열을 합침

    return result; // 왼쪽 정렬된 텍스트를 반환
}
```

9.25 오른쪽_정렬() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------
메소드명: 오른쪽_정렬
기능: 주어진 텍스트를 오른쪽 정렬하는 함수로, 오른쪽 정렬된 텍스트를 반환함
파라미터: text
반환값: 오른쪽 정렬된 텍스트
—----------------------------------------------------------------------------------------*
public String alignRight(String text) {
    // 함수 정의: 주어진 텍스트를 오른쪽 정렬함
    int screenWidth = 80; // 화면 너비를 예시로 80으로 설정
      int paddingSize = screenWidth - text.length(); // 패딩 사이즈를 계산 (화면 너비에서 텍스트 길이를 뺌)
```

```
    if (paddingSize < 0) { // 패딩 사이즈가 음수일 경우 (텍스트가 화면 너비보다 긴 경우)
        return text; // 텍스트를 그대로 반환
    }

    String padding = String.format("%" + paddingSize + "s", ""); // 패딩 문자열 생성 (공백으로 채움)
    String result = padding + text; // 패딩 문자열과 텍스트를 합침

    return result; // 오른쪽 정렬된 텍스트를 반환
}
```

9.26 쪽번호_매기기() 알고리즘 명세

*----------

메소드명: 쪽번호_매기기

기능: 주어진 전체 페이지 수에 대해 각 페이지에 쪽 번호를 매기는 함수로, 각 페이지에 대해 페이지 번호를 입력하여 페이지에 쪽 번호를 매김

파라미터: totalPageCount

반환값: 없음 (output이 없는 함수)

—----------*

```
public void assignPageNumbers(int totalPageCount) {
    // 함수 정의: 주어진 전체 페이지 수에 대해 각 페이지에 쪽 번호를 매김
```

```
    for (int i = 1; i <= totalPageCount; i++) { // 전체 페이지 수만큼 반복

        String pageNumber = "Page " + i; // 페이지 번호 생성

            printPageNumber(pageNumber); // 페이지에 쪽 번호를 출력하는 함수 호출 (이 함수는 실제로는 페이지에 쪽 번호를 매기는 로직을 구현해야 함)

    }

}
```

9.27 그림_배치() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------

메소드명: 그림_배치

기능: 전체 그림을 화면에 배치하는 함수로, 화면 가로 길이와 세로 길이에 따라 그림의 위치를 계산하여 배치 정보를 반환함

파라미터: totalPictures, screenWidth, screenHeight

반환값: 그림 배치 정보를 나타내는 리스트

—----------------------------------------------------------------------------------------*
public List<Integer> arrangePictures(int totalPictures, int screenWidth, int screenHeight)
{

    // 함수 정의: 전체 그림을 화면에 배치함



      List<Integer> pictureArrangementInfo = new ArrayList<>(); // 각 그림의 배치 정보를 저장할 리스트 생성



    for (int i = 0; i < totalPictures; i++) { // 전체 그림 수만큼 반복

         int picturePosition = screenHeight / (totalPictures + 1) * (i + 1); // 화면 세로 길이에 따라 그림 위치 계산

        pictureArrangementInfo.add(picturePosition); // 계산된 그림 위치를 그림 배치 정보에
```

추가

}

return pictureArrangementInfo; // 그림 배치 정보를 반환

}

9.28 문단_배치() 알고리즘 명세

*----------

메소드명: 문단_배치

기능: 전체 문단을 화면의 세로 길이에 따라 나누어 배치하는 함수로, 화면 가로 길이, 화면 세로 길이, 및 간격을 고려하여 각 문단의 배치 정보를 계산함

파라미터: totalParagraphs , screenWidth, screenHeight, spacing

반환값: 문단 배치 정보를 나타내는 리스트

—----------*

```
public List<Integer> arrangeParagraphs(int totalParagraphs, int screenWidth, int screenHeight, int spacing) { // 함수 정의

    List<Integer> paragraphArrangementInfo = new ArrayList<>(); // 각 문단의 배치 정보를 저장할 리스트 생성


  for (int i = 0; i < totalParagraphs; i++) { // 전체 문단 수만큼 반복

        int paragraphPosition = screenHeight / (totalParagraphs + 1) * (i + 1); // 화면 세로 길이에 따라 문단 위치 계산

      paragraphArrangementInfo.add(paragraphPosition); // 계산된 문단 위치를 문단
```

```
배치 정보에 추가

  }



  return paragraphArrangementInfo; // 문단 배치 정보를 반환

}
```

9.29 그림책_편집_요청() 알고리즘 명세

```
*----------------------------------------------------------------------------------------
```

메소드명: 그림책_편집_요청

기능: 주어진 그림책 정보를 사용자의 편집 요청에 따라 수정하는 함수로, 텍스트 수정 및 그림 추가 등 다양한 편집 유형을 처리함

파라미터: pictureBookInfo, userEditRequest

반환값: 편집 결과를 나타내는 문자열 ("편집이 완료되었습니다.")

```
—----------------------------------------------------------------------------------------*
public String saveParagraphs(List<String> paragraphList, String saveLocation) {

  // 함수 정의: 주어진 문단 목록을 파일 또는 데이터베이스에 저장함



  if (saveLocation is a file path) { // 저장 위치가 파일 경로인 경우

    try {

      FileWriter fileWriter = new FileWriter(saveLocation); // 파일 작성자 생성

      for (String paragraph : paragraphList) { // 문단 목록의 각 문단에 대해

        fileWriter.write(paragraph); // 문단을 파일에 씀

      }
```

```
        fileWriter.close(); // 파일 작성자를 닫음

          return "Paragraphs have been successfully saved in the file."; // 성공 메시지 반환

      } catch (IOException e) {

         return "An error occurred while saving the paragraphs in the file."; // 파일 저장 중 오류 발생시 오류 메시지 반환

      }

    } else if (saveLocation is a database connection) { // 저장 위치가 데이터베이스 연결인 경우

      try {

                                    DatabaseConnection dbConnection = new DatabaseConnection(saveLocation); // 데이터베이스 연결 생성

        for (String paragraph : paragraphList) { // 문단 목록의 각 문단에 대해

          dbConnection.save(paragraph); // 문단을 데이터베이스에 저장

        }

          return "Paragraphs have been successfully saved in the database."; // 성공 메시지 반환

      } catch (DatabaseException e) {

          return "An error occurred while saving the paragraphs in the database."; // 데이터베이스 저장 중 오류 발생시 오류 메시지 반환

      }

   } else {

      return "The save location is not valid."; // 저장 위치가 올바르지 않은 경우 오류 메시지 반환

   }
```

```
}
```

9.30 스토리_저장() 알고리즘 명세

```
*-----------------------------------------------------------------------------------------
```

메소드명: 스토리_저장

기능: 주어진 스토리 내용을 파일에 저장하거나 데이터베이스에 저장하는 함수로, 입력된 파일 경로 또는 데이터베이스 연결을 기반으로 동작함

파라미터: storyContent, saveLocation

반환값: 저장 결과를 나타내는 문자열 (저장 성공 또는 실패 메시지)

```
—----------------------------------------------------------------------------------------*
public String saveStory(String storyContent, String saveLocation) {
   // 함수 정의: 주어진 스토리 내용을 파일 또는 데이터베이스에 저장함


   if (saveLocation is a file path) { // 저장 위치가 파일 경로인 경우
      try {
         FileWriter fileWriter = new FileWriter(saveLocation); // 파일 작성자 생성
         fileWriter.write(storyContent); // 스토리 내용을 파일에 씀
         fileWriter.close(); // 파일 작성자를 닫음
         return "Story has been successfully saved in the file."; // 성공 메시지 반환
      } catch (IOException e) {
            return "An error occurred while saving the story in the file."; // 파일 저장 중 오류 발생시 오류 메시지 반환
      }
    } else if (saveLocation is a database connection) { // 저장 위치가 데이터베이스
```

```
연결인 경우

    try {

                                    DatabaseConnection dbConnection = new
DatabaseConnection(saveLocation); // 데이터베이스 연결 생성

        dbConnection.save(storyContent); // 스토리 내용을 데이터베이스에 저장

            return "Story has been successfully saved in the database."; // 성공 메시지
반환

    } catch (DatabaseException e) {

                return "An error occurred while saving the story in the database."; //
데이터베이스 저장 중 오류 발생시 오류 메시지 반환

    }

  } else {

      return "The save location is not valid."; // 저장 위치가 올바르지 않은 경우 오류
메시지 반환

  }

}
```

9.31 문단_저장() 알고리즘 명세

*----------

메소드명: **save_paragraphs**

기능: 주어진 나눈 문단 목록을 파일에 저장하거나 데이터베이스에 저장하는 함수로, 입력된 파일 경로 또는 데이터베이스 연결을 기반으로 동작함

파라미터: paragraphList , saveLocation

반환값: 저장 결과를 나타내는 문자열 (저장 실패시, 오류 메시지)

```
—----------------------------------------------------------------------------------------*
public String saveParagraphs(List<String> paragraphList, String saveLocation) {
  // 함수 정의: 주어진 문단 목록을 파일 또는 데이터베이스에 저장함

  if (saveLocation is a file path) { // 저장 위치가 파일 경로인 경우
    try {
      FileWriter fileWriter = new FileWriter(saveLocation); // 파일 작성자 생성
      for (String paragraph : paragraphList) { // 문단 목록의 각 문단에 대해
        fileWriter.write(paragraph); // 문단을 파일에 씀
        fileWriter.write(System.lineSeparator()); // 각 문단 사이에 줄바꿈 추가
      }
      fileWriter.close(); // 파일 작성자를 닫음
        return "Paragraphs have been successfully saved in the file."; // 성공 메시지 반환
    } catch (IOException e) {
       return "An error occurred while saving the paragraphs in the file."; // 파일 저장 중 오류 발생시 오류 메시지 반환
    }
   } else if (saveLocation is a database connection) { // 저장 위치가 데이터베이스 연결인 경우
    try {
                                DatabaseConnection dbConnection = new DatabaseConnection(saveLocation); // 데이터베이스 연결 생성
      for (String paragraph : paragraphList) { // 문단 목록의 각 문단에 대해
```

```
            dbConnection.save(paragraph); // 문단을 데이터베이스에 저장

        }

            return "Paragraphs have been successfully saved in the database."; // 성공 메시지 반환

    } catch (DatabaseException e) {

            return "An error occurred while saving the paragraphs in the database."; // 데이터베이스 저장 중 오류 발생시 오류 메시지 반환

    }

  } else {

      return "The save location is not valid."; // 저장 위치가 올바르지 않은 경우 오류 메시지 반환

  }

}
```

10. 요구사항 검토 결과

(해당 사항 없음)

11. 결론

11.1 부록

(해당 사항 없음)

11.2 참고문헌

11조_산학프로젝트_시스템 제안서.pdf

11조_프로젝트 요약서.pdf