Pusan National University
Computer Science and Engineering
Technical Report 2025-09

어린이가 작성한 이야기(일기) 기반 그림 동화 생성기

201224544 차행철

201924511 오지웅

202055610 최광진

지도교수 탁성우

목 차

1. 서론

1.1. 연구 배경

일기 쓰기는 학생들이 글쓰기 능력을 기르고, 자신의 경험을 서술하며, 더 나아가 자아를 이해하고 미래의 자신에게 소중한 추억을 전달하는 데 중요한 교육적 의미를 지닌다. 그러나 일부 학생들에게 일기 작성은 지루하고 부담스러운 과제로 인식되기도 하며, 이러한 이유로 학습 효과가 저하되는 경우가 있다. 반대로 일기 쓰기를 즐기는 학생들조차도 자신이 기록한 일기를 더욱 생생하게 표현하거나, 등장인물과 사건을 확장하여 상상력을 발휘하고 싶어 하는 경우가 많다.

따라서 일기를 바탕으로 그림 동화 이야기를 자동으로 생성해 주는 애플리케이션은 학생들에게 새로운 동기를 부여할 수 있다. 일기를 쓰기 힘들어하는 학생들에게는 즐겁게 글쓰기에 접근할 수 있는 계기를 제공하고, 일기를 좋아하는 학생들에게는 자신의 글을 구체화하고 확장할 수 있는 창의적 경험을 제공한다. 이러한 과정은 단순한 기록 활동을 넘어, 개인의 일상과 상상이 결합된 새로운 형태의 이야기 창작 경험으로 이어질 수 있다.

나아가 본 연구는 단순히 동화를 생성하는 데 그치지 않고, 인공지능(AI)을 활용하는 과정에서 필요한 프롬프트(prompt) 작성 기능을 학생들이 직접 경험할 수 있도록 기능을 확장하고자 한다. 이를 통해 학생들은 AI 시대에 요구되는 핵심 역량 중 하나인 대규모 언어 모델(LLM) 활용 능력을 자연스럽게 익히고, 보다 유창하고 주도적으로 AI와 상호작용할 수 있는 능력을 기를 수 있다.

1.2. 기존 문제점

최근 아이들을 대상으로 한 AI 기반 이야기 생성 서비스가 다수 개발되어 있다. 본 연구에서는 기존 서비스들의 특징과 한계를 분석하여 개발 방향성을 도출하고자 한다.

1) 모두의 동화

주요 기능: 사용자가 직접 이야기의 내용을 작성하고, 표지를 꾸미며, 완성된 이야기를 다른 사용자와 공유할 수 있는 플랫폼 제공한다.
장점: 사용자 주도적 창작 경험 제공, 공동체 공유를 통한 성취감 부여
단점: 이야기 생성이 전적으로 사용자에게 의존하므로, 부적절하거나 편향된 내용이 포함될 가능성이 있다. AI 기반 검증 기능이 없어 관리가 어렵다는 문제가 제기됨.

2) Storybook AI

주요 기능: 생성형 AI를 기반으로 그림 동화를 생성하며, 문장 단위로 사용자가 내용을 확인(confirm)하고 피드백을 제공할 수 있음.
장점: 사용자가 원하는 내용을 적극적으로 반영할 수 있어 맞춤형 이야기 생성 가능. 즉각적 피드백을 통해 만족도 향상.
단점: 주 이용자가 아동이라는 특성으로 인해, 이야기가 길어지면 집중력이 떨어지고 흥미를 쉽게 잃음. 문장 단위 확인 과정이 오히려 사용자의 몰입을 방해할 수 있음.

3) Storybookly

주요 기능: 생성형 AI 기반의 그림 동화 웹 서비스로, 부드러운 색감과 친숙한 그림체 제공.

장점: 시각적 친밀감이 높아 아동의 흥미 유발에 효과적임.
단점: 배경음악이나 사운드 효과가 제공되지 않아 몰입 경험이 제한됨. 사용자 경험 향상을 위한 멀티모달 요소 부족.

종합적으로 살펴보았을 때, 기존 서비스들은 각기 차별화된 장점을 가지고 있으나, 아동 사용자 특성과 AI 기반 생성 과정에서 발생할 수 있는 문제점을 충분히 보완하지 못하고 있다. 특히 사용자 주도형 서비스에서는 부적절한 콘텐츠 발생 가능성이 높고, 생성 과정에서 반복 확인이 요구되는 서비스는 집중력 저하와 흥미 상실 문제를 야기한다. 또한 몰입 경험을 강화할 수 있는 멀티모달 요소(배경음악, 사운드 등)의 부재도 공통된 한계로 나타난다.

1.3. 연구 목표

본 연구의 목표는 아동이 자신의 일기를 바탕으로 그림 동화 이야기를 생성하는 과정에서, 사용자 경험, 콘텐츠 생성, 학습·심리적 효과를 종합적으로 고려한 애플리케이션을 개발하는 것이다.

1) 사용자 경험(UX) 향상

인터페이스는 직관적이고 간단하게 설계하여, 일기 작성, 촬영, 업로드 등 일련의 과정이 원활히 진행되도록 한다.

비동기 처리와 즉각적 응답을 지원하여, 파이프라인 작업 중에도 사용자가 다른 활동을 수행할 수 있도록 한다.

작업 완료 시 푸시 알림을 제공하여, 사용자가 진행 상태를 쉽게 확인할 수 있도록 한다.

전체 애플리케이션은 빠른 처리 속도와 높은 반응성을 제공하여 아동의 집중력과 흥미를 유지한다.

2) 감정 기반 이야기 생성

사용자의 입력에서 드러나는 감정과 핵심 요소를 최대한 활용하여 이야기를 생성한다.

부적절한 내용은 필터링하되, 부정적 감정이나 특징적 요소는 적절히 반영하여 사용자 경험과 심리적 진정성을 유지한다.

사용자의 감정은 그림으로 시각화되며, 교육자와 보호자가 피드백을 제공할 수 있도록 한다.

3) 학습적·심리적 효과 강화

일기와 감정을 기반으로 이야기를 생성하면서, 새로운 시각과 객관적 관점을 제공한다.

이를 통해 사용자가 다양한 관점에서 상황을 이해하고, 사고력과 상상력을 확장할 수 있도록 한다.

호기심을 자극하는 요소를 적극 반영하여 학습 동기를 강화하고, 이야기 경험의 몰입도를 높인다.

2. 연구 배경

2.1. Ai 서버에서 사용한 기술

2.1.1. hugging face

- ○ nlp, ml 관련 연구와 생태계를 만든 회사
- ○ hugging face hub, spaces, inference api 등 서비스를 제공
- ○ hugging face hub: 모델, 데이터셋, 토크나이저 등 공유 플랫폼. 오픈소스 모델, 모델 가중치, 데이터셋을 업로드/다운로드 할 수 있음
- ○ inference api: 서버리스 추론 기능. 무료버전에는 토큰 제한 있음

2.1.2. transformer

- ○ 사전학습된 트랜스포머 모델 저장/추론/재학습/미세조정 등을할 수 있는 범용 프레임워크
- ○ hugging face hub 에서 모델 아키텍처와 사전 학습된 가중치 다운로드하기 위해 사용
- ○ 입력된 문장을 토큰 단위로 분해하는 토크나이저 기능 제공

2.1.3. sentence transformer

- ○ hugging face 모델을 기반으로 문장단위 임베딩에 특화되어 만들어진 프레임워크
- ○ cosin similarity 기반 검색, semantic search, clustering 등 문장 단위 작업을 쉽게 할 수 있는 유틸 함수 제공

2.1.4. Helsinki-NLP/opus-mt-ko-en

- ○ Hugging face hub 링크: https://huggingface.co/Helsinki-NLP/opus-mt-ko-en
- ○ 원본 저장소: https://github.com/Helsinki-NLP/Tatoeba-Challenge/blob/master/models/kor-eng/README.md
- ○ 순정 모델: https://object.pouta.csc.fi/Tatoeba-MT-models/kor-eng/opus-2020-06-17.zip
- ○ 테스트셋: https://object.pouta.csc.fi/Tatoeba-MT-models/kor-eng/opus-2020-06-17.test.txt
  (테스트 셋이 깨져 보이는 것은 디코딩 방식 때문이고 원본은 한글)
- ○ 언어구성: Korean -> English
- ○ 모델타입: 단일언어 -> 단일언어
- ○ chrF2: 0.588, BLEU: 41.3, Brevity Penalty: 0.959
- ○ 사용량: 지난달 261k

특징: 가장 보편적으로 사용되는 오픈소스 한영 번역 모델

2.1.5. NHNDQ/nllb-finetuned-en2ko

- ○ Hugging face hub : https://huggingface.co/NHNDQ/nllb-finetuned-en2ko
- ○ 테스트셋: https://www.aihub.or.kr/
- ○ BLEU: 33.66
- ○ 특징: 자연어 처리 전문 업체인 nhn 다이퀘스트 개발.
  fine tuning 을 통한 원본 모델 대비 BLEU +11

2.1.6. llama(large language model meta ai)

- ○ Meta ai 에서 개발한 대규모 언어 모델 시리즈
- ○ Transformer 기반 대규모 언어 모델
- ○ 사용자 요구에 맞는 다양한 모델 크기 지원
- ○ 사전 학습된 모델 가중치, 추론툴, ollama, 통신용 api 등 편의기능 제공

2.1.7. Ollama

- ○ 로컬 환경에서 llama 계열 같은 대규모 언어 모델을 쉽게 실행하기 위한 플랫폼
- ○ Mac, linux, window 등 운영체제 지원, Gpu 없이 cpu 기반 실행 가능
- ○ 모델 실행, 다운로드, 관리용 명령어 제공
- ○ Hugging face 처럼 모델 공유 허브 제공을 통해 쉽게 가중치 접근

2.1.8. Llama 3.2b:3G

- ○ 3.2b 개의 파라미터를 가짐. 비교군으로 gpt3 (175b) 이 있음
- ○ Llama 3 계열에서 작은 크기의 모델을 의미
- ○ 선형계층: FP16 에서 int4 로 4-bit groupies scheme quantization
- ○ 활성화: 8bit per token dynamic quantization
- ○ 분류 레이어: 8 bit per channel quantization
- ○ 분류 레이어 활성화: 8 bit per token dynamic quantization
- ○ 임베딩 레이어: 8 bit per channel quantization

2.1.9. nreimers/MiniLM-L6-H384-uncased

- ○ 10 억+ 문장 쌍을 활용한 **Sentence Embedding** 학습
- ○ Self-supervised contrastive learning (문장 쌍 매칭 학습)
- ○ 문장/문단을 **벡터 임베딩**으로 변환하여 정보 검색, 클러스터링, 문장 유사도 계산
- ○ 입력 길이: 최대 256 토큰

2.1.10. stabilityai/stable-diffusion-xl-base-1.0

- ○ **개발 주체**: Stability AI
- ○ **모델 유형**: 텍스트 기반 이미지 생성 모델 (Latent Diffusion Model)
- ○ **Base Model**: 초기 이미지(latent) 생성
- ○ **Refiner Model**: 고해상도·고품질 출력을 위해 최종 노이즈 제거
- ○ **텍스트 인코더**: OpenCLIP-ViT/G, CLIP-ViT/L (사전학습 모델 고정 사용)
- ○ 텍스트 프롬프트 기반 이미지 생성 및 수정
- ○ GitHub: Stability-AI/generative-models

2.1.11. Redis queue

- ○ Redis Queue(RQ)는 **Python 기반의 간단한 작업 큐(job queue)** 라이브러리
- ○ **Redis**를 메시지 브로커(message broker)로 사용하여, 백그라운드에서 비동기 작업을 실행할 수 있게 함
- ○ Celery와 같은 복잡한 프레임워크보다 **경량화된 대안**으로, 빠른 개발 및 간단한 아키텍처를 지향

2.2. 백엔드 서버에서 사용한 기술

2.2.1. 개발 환경 및 프레임워크

1. **프로그래밍 언어 및 프레임워크**
   - ○ 본 프로젝트의 백엔드 서버는 **Java 21**을 기반으로 개발되었으며, **Spring Boot 3.x** 프레임워크를 활용하여 웹 애플리케이션 및 API 서버를 구현하였다.
   - ○ Spring Boot의 의존성 주입, 자동 설정(Auto-Configuration), REST API 지원 기능을 활용하여 **개발 생산성 향상**과 **유지보수 용이성**을 동시에 확보하였다.
2. **빌드 도구 및 설정 관리**
   - ○ 프로젝트 빌드와 의존성 관리는 **Gradle(Groovy DSL)**을 사용하였다.
   - ○ 서버 환경 설정은 `application.properties` 파일을 통해 일관되게 관리되며, 기본 패키지 네임은 `com.wudc.storypool`로 설정하였다.

이러한 구조는 프로젝트의 **모듈화와 확장성**을 높이는 동시에, 팀 협업 환경에서의 코드 관리 효율성을 강화한다.

2.2.2. 외부 서비스 연동

1. **푸시 알림 서비스**
    - ○ 모바일 및 웹 클라이언트와의 실시간 커뮤니케이션을 위해 **Firebase Cloud Messaging(FCM)**을 사용하였다.
    - ○ 이를 통해 서버에서 클라이언트로 즉시 알림을 전달할 수 있으며, 사용자 경험을 향상시키는 핵심 기능으로 활용되었다.
2. **이메일 전송 서비스**
    - ○ 사용자 인증 및 알림을 위해 **Gmail SMTP** 를 연동하였다.
    - ○ 안정적이고 표준화된 SMTP 프로토콜을 기반으로 하여, 이메일 전송의 신뢰성과 보안성을 확보하였다.
3. **파일 저장 및 관리**
    - ○ 이미지, 문서 등 대용량 파일 관리를 위해 **AWS S3** 를 활용하였다.

클라우드 기반 객체 저장소를 사용함으로써, 서버 부하를 최소화하고, 데이터의 **내구성과 접근성을 보장**하였다.

2.2.3. 서버 인프라 및 운영 환경

1. **도메인 관리 및 네트워크 보안**
    - ○ 서버 도메인 및 DNS 관리는 **AWS Route 53** 을 사용하여 안정적인 도메인 라우팅과 트래픽 관리 기능을 제공하였다.
    - ○ **Cloudflare Tunneling** 을 활용하여 외부 공격으로부터 서버를 보호하고, 안정적인 네트워크 접속 환경을 구성하였다.
2. **웹 서버 및 리버스 프록시**
    - ○ **Nginx** 를 기반으로 HTTPS 및 SSL/TLS 리버스 프록시 환경을 구성하여, **데이터 전송 보안**과 **트래픽 분산 처리**를 지원하였다.
3. **컨테이너 환경 및 배포**
    - ○ 서버 환경을 **Docker 컨테이너**로 구성하여, 배포 자동화와 환경 일관성을 확보하였다.
    - ○ 컨테이너 기반 배포는 서버 환경의 **독립성 확보**와 **신속한 확장성**에 기여하였다.

4. **데이터베이스 및 캐시 시스템**
    - ○ 핵심 데이터 관리를 위해 **MySQL** 을 사용하였으며, 구조화된 관계형 데이터베이스 설계를 통해 **데이터 무결성과 일관성**을 보장하였다.
    - ○ **Redis** 를 캐시 시스템으로 활용하여, 반복적인 데이터 조회 속도를 향상시키고 서버 성능을 최적화하였다.

2.2.4. 기술 선택의 의의

- ○ 본 백엔드 서버 아키텍처는 **확장성, 안정성, 보안성**을 종합적으로 고려하여 설계되었다.
- ○ 다양한 외부 서비스와의 연동을 통해 사용자 경험을 극대화하고, 컨테이너 기반 환경과 클라우드 인프라 활용으로 **운영 효율성**을 강화하였다.
- ○ 또한, 현대 웹 애플리케이션의 요구사항에 부합하는 **모듈화된 설계**와 **표준화된 기술 스택**을 적용함으로써, 향후 유지보수와 기능 확장에 유리한 구조를 확보하였다.

2.3. 클라이언트 앱에서 사용한 기술

2.3.1. Google ML kit for OCR

- ○ 사용자가 작성한 일기나 짧은 글을 그림동화로 생성해야 한다.
- ○ 이를 위해 앱을 통해 직접 텍스트를 입력 받을수도 있지만, 사용자가 작성한 손글씨를 직접 촬영한 뒤 텍스트를 추출하여 어플리케이션에 넘겨주는 기능이 필요하다.
- ○ 이를 위해 OCR 기능을 어플리케이션에 추가하였다.
  https://developers.google.com/ml-kit/vision/text-recognition/v2/android?hl=ko

2.3.2. Unity

- ○ 사용자가 동화를 쉽게 생성하고 다른 사람이 생성한 동화를 감상할 수 있는 UI/UX 가 제공되어야 한다.
- ○ 이를 위해 Unity 에서 제공되는 UI components 들을 활용하고 클라이언트 로직을 구성하여 화면 흐름을 설계하였다.
- ○ 사용한 버전은 6000.0.44f1 이다.

3. 연구 내용

3.1. 인공지능 서버 개발

3.1.1. 비동기 처리

본 시스템은 효율적인 병렬 처리와 사용자 경험 향상을 위해 **Redis 기반 작업 큐와 워커(Worker) 구조**를 채택하였다. 웹 서버는 사용자로부터 작업 요청을 수신하면 해당 작업을 **큐에 등록(enqueue)** 하고 즉시 완료 응답을 반환한다. 큐에는 수행해야 할 작업의 순서와 필요한 입력 데이터가 기록되며, 워커는 유휴 상태에서 큐로부터 작업을 가져와 실행한다. 모든 파이프라인 단계가 완료되면 최종 결과가 웹 서버로 전달되어 사용자에게 제공된다.

3.1.2. Scene Parsing

생성된 이야기 데이터를 기반으로 **대규모 언어 모델(LLM)**을 활용하여 씬 단위의 구조적 파싱을 수행한다.이 과정에서는 **format-only one-shot 프롬프트**를 사용하며, 결과가 **JSON 형식**을 준수하는지 검증하고, 필요 시 재시도 로직을 적용한다. 또한, 이야기 누락을 방지하기 위해 각 씬별로 **missing part 보정 로직**을 수행하여 데이터 완전성을 확보한다.

3.1.3. Prompt Maker

Scene Parsing 결과를 기반으로 각 씬의 **이미지 생성용 프롬프트**를 구성한다. 초기 단계에서는 등장인물의 외형, 의상 등 핵심 요소 누락 및 프롬프트 일관성 저하 문제가 발생하였다. 이를 개선하기 위해 각 씬별로 등장인물의 외형, 의상, 소품, 요소 간 상호작용을 개별적으로 생성하고, 이를 종합하여 최종 프롬프트를 완성한다. 이를 통해 일관성과 정밀도를 확보한 이미지 생성을 지원한다.

3.2. 백엔드 서버 개발

3.2.1. 인프라 설계

- ○ **FCM(Firebase Cloud Messaging)**: 실시간 푸시 알림 전송을 위한 서비스로, 사용자 기기와 서버 간 즉각적인 알림 전달을 가능하게 함.
- ○ **Gmail SMTP**: 사용자 인증, 비밀번호 초기화, 알림 메일 발송 등 이메일 기반

기능을 제공하기 위해 SMTP 프로토콜 기반으로 연동.

- **AWS S3**: 이미지 및 파일 업로드를 위한 객체 스토리지로, Presigned URL 방식을 통해 서버 부하를 최소화하고 안전한 업로드 경로를 제공.
- **AWS Route 53**: 도메인 및 DNS 관리를 담당하여 안정적인 네트워크 라우팅을 보장.
- **Cloudflare Tunneling**: 외부 공격 방어와 안정적 접속 환경 제공을 위해 적용.
- **Docker**: 서버 배포 환경을 컨테이너 단위로 관리하여 이식성과 일관성을 확보.

3.2.2. 패키지 구조 설계

본 프로젝트는 **도메인 중심 구조(Domain-Oriented Architecture)**를 적용하였으며, 주요 패키지는 다음과 같이 구성되었다.

- **common**: 전역적으로 사용되는 BaseEntity, 예외 처리, 공통 유틸리티 클래스 포함.
- **domain**:
  - `user`: 사용자 인증, 프로필 관리, 이메일 인증 및 JWT 기반 보안 처리.
  - `story`: 스토리 초안 생성, 수정, 삭제 및 조회 기능 제공.
  - `fairytale`: AI 동화 생성 및 상태 관리.
  - `post`: 게시글 CRUD 및 좋아요 기능.
  - `comment`: 댓글 및 대댓글 관리, 좋아요 기능.
  - `notification`: 알림 전송, 읽음 처리, 알림 설정 관리.
  - `upload`: AWS S3 Presigned URL 발급 및 파일 업로드.
- **global**: 전역 Config(Spring Security, Redis, Firebase, Swagger, S3 등), 인터셉터, 헬스체크 API 제공.

security: JWT 토큰 발급 및 검증, 인증·인가 로직, 필터 처리.

3.2.3. API 설계

본 백엔드 서버는 **RESTful API**를 기반으로 하며, Swagger(OpenAPI 3.1)를 통해 문서화하였다. 주요 API 그룹은 다음과 같다.

1. **사용자 인증 및 관리(Auth, User)**
   - 회원가입, 로그인, 로그아웃, 토큰 갱신, 비밀번호 초기화, 회원 탈퇴 지원.
   - 이메일 인증 코드 발송 및 검증.

○ 사용자 프로필 조회/수정 API 제공.

2. **스토리 및 동화 관리(Story, Fairytale)**
   ○ 사용자 스토리 초안 작성, 수정, 삭제, 조회.
   ○ AI 기반 동화 생성, 상태 조회 및 관리.
3. **커뮤니티 기능(Post, Comment)**
   ○ 게시글 CRUD 및 좋아요 기능.
   ○ 댓글 및 대댓글 CRUD, 좋아요 기능.
4. **알림(Notification, Device)**
   ○ 디바이스 등록 및 해제.
   ○ 푸시 알림 목록 조회, 읽음 처리, 알림 삭제.
   ○ 알림 설정 조회 및 수정.
5. **파일 업로드(Upload)**
   ○ AWS S3 Presigned URL 발급.
   ○ 클라이언트에서 직접 파일 업로드 가능하도록 설계.
6. **기타**
   ○ `/health` API 를 통한 서버 상태 체크.
   ○ Swagger UI 를 통한 API 문서 접근 제공.

3.2.4. 인증 및 보안

○ **JWT(Json Web Token)** 기반 인증 시스템 적용.
○ Access Token 과 Refresh Token 을 사용하여 보안성과 사용자 경험을 동시에 확보.
○ Spring Security 를 기반으로 한 필터 체계(JwtAuthFilter, SecurityExceptionFilter)를 통해 인증·인가 절차 수행.
○ 사용자 비밀번호는 암호화하여 저장하며, 인증 코드 기반 이메일 검증 로직 적용.

3.2.5. 데이터 관리

○ **MySQL**: 관계형 데이터베이스로 사용자, 스토리, 게시글, 댓글, 알림 등 주요 도메인 데이터 관리.
○ **Redis**: 토큰 저장, 캐싱, 인증 시도 제한 등 세션 관리에 활용하여 응답 속도와 보안성 강화.

3.2.6. DB 설계

본 프로젝트의 데이터베이스는 MySQL 기반으로 설계되었으며, JPA/Hibernate를 통해 객체-관계 매핑을 적용하였다. 핵심 엔티티는 다음과 같이 구성된다.

1. 사용자(User) 및 기기(Device) 관리

- **User**: 이메일, 비밀번호, 닉네임, 프로필 이미지, 권한(Role) 등을 관리한다.
  - `@SQLDelete` 및 `@SQLRestriction`을 활용하여 소프트 삭제(Soft Delete) 방식 적용.
  - `@OneToMany` 관계를 통해 다수의 **Device**와 연결된다.
- **Device**: 사용자별 기기 정보를 관리하며, FCM 토큰과 플랫폼(Android/iOS)을 저장한다.
  - `(user_id, deviceId)`에 대한 Unique 제약을 두어 중복 등록을 방지한다.

2. 인증 및 세션 관리

- **Auth (RedisHash)**: 이메일 인증코드 및 만료 시간을 Redis 기반으로 관리한다.
  - `@TimeToLive` 어노테이션을 활용하여 TTL 기반 자동 만료 처리.

3. 창작물(Story, Fairytale) 관리

- **Story**: 사용자가 작성하는 기본 이야기를 저장하며, 수정 및 삭제 기능을 제공한다.
- **Fairytale**: Story를 기반으로 확장 생성된 동화를 관리한다.
  - 상태(`FairytaleStatus`), 페이지 수, 메시지 등을 기록.
  - `@OneToMany` 관계로 **FairytaleePage**를 보유하며, 동화의 각 페이지(스토리·이미지)를 저장한다.

3.2.6.1. 4. 게시물(Post) 및 상호작용

- **Post**: 동화 기반으로 작성된 게시물, 태그, 조회수·좋아요·댓글 수를 관리한다.
- **Like**: 게시물 좋아요를 기록하며 `(userId, postId)`에 Unique 제약을 두었다.
- **Comment**: 게시물 댓글 및 대댓글을 관리한다.
  - `parentId` 필드를 통해 계층 구조를 구현.
  - 좋아요 수, 대댓글 수를 별도로 관리한다.
- **CommentLike**: 댓글 좋아요를 기록하며 `(userId, commentId)`에 Unique 제약을 둔다.

3.2.6.2. 5. 알림(Notification) 관리

- **NotificationSettings**: 사용자별 푸시/이메일 알림 설정을 관리한다.
- **Notification**: 댓글, 좋아요, 동화 생성 완료 등의 이벤트를 사용자에게 전달하기 위한 알림 정보를 저장한다.
  - `(user_id, createdAt, id)` 인덱스를 활용하여 효율적인 조회를 지원.

3.3. 클라이언트 앱 개발

3.3.1. Google ML kit for OCR

- 이미지 입력
  사용자는 카메라 또는 갤러리에서 촬영한 손글씨 이미지를 업로드한다.
- OCR 처리
  ML Kit 의 API 를 통해 이미지 내의 텍스트를 분석하고, 단어 단위로 인식된 결과를 문자열로 변환한다.
- 결과 활용
  추출한 문자열을 서버를 통해 AI 모델로 전송하여 그림 동화로 변환하는 데 활용한다. 문장에 대한 전처리는 AI 모델에서 수행한다.

- Unity 에서 사용자의 카메라를 사용하기 위해 Native Camera for Android & iOS 라이브러리를 사용하였다. 또한, 이미지의 Exif 정보를 받아 오기 위해 ExifLib 을 사용하여 이미지의 정보를 받아와 클라이언트에서 처리할 수 있게 하였다.

3.3.2. Unity

○ 사용자 편의를 위한 UI/UX 제공

Unity에서 제공되는 Button, Canvas, TextMeshPro와 같은 UI Components를 사용하여 직관적인 화면을 구성한다. 또한 사용자가 앱의 사용 방식을 쉽게 알 수 있도록 단순하면서도 직관적인 상호작용을 작성하였다.

○ 서버에서 제공되는 API 를 클라이언트에서 호출하여 AI 모델로 요청

UnityWebRequest을 활용하여 서버에서 제공되는 API를 비동기방식으로 호출한다. 동화 생성, 관리와 관련된 연산은 모두 서버에서 처리되고 클라이언트에서는 결과만 받아 화면에 표시한다. 비동기 처리를 순차적으로 처리하기 위해 C#의 Coroutine을 사용하였다.

○ 생성된 동화를 앱에서 조회 및 동화 분위기에 맞는 배경음악 재생

서버에서 이미지와 텍스트를 받아와 화면에 표시한다. 또한 동화의 각 장면마다 저장되어있는 감정(분위기)를 기반으로 적절한 배경음악을 같이 제공하여 사용자가 더욱 동화에 몰입할 수 있는 환경을 제공한다.

○ 시각효과 제공

사용자가 어플리케이션을 사용하거나, 동화를 감상할 때 좀 더 쉽게 몰입할 수 있도록 Unity의 Animation component를 활용한 다양한 형태의 시각효과를 제공하였다.

4. 연구 결과 분석 및 평가

4.1. Prompt Making 평가

본 연구에서는 **Prompt Making 과정에서 생성된 프롬프트의 품질**을 정성적 요소를 기반으로 정량적으로 평가하였다. 이를 통해 프롬프트 생성 과정의 개선점을 도출하고, 개선된 Prompt Maker의 성능을 검증하는 데 활용하였다.

4.1.1. 평가 목적

○ 프롬프트 제작 과정에서 **등장인물, 장소, 오브젝트, 관계성, 그림체 일관성** 등의 요소가 적절히 반영되었는지 평가.

○ 평가 결과를 바탕으로 프롬프트 생성 과정의 **일관성 및 완전성**을 개선.

4.1.2. 평가 지표 및 방법

○ 각 스토리에 대해 Scene 단위로 프롬프트를 생성하고, **정성적 기준**을 0~3 점 척도로 정량화하여 평가.

○ 평가 항목:

1. **등장인물 묘사 일관성**: 모든 장면에서 핵심 인물 묘사가 일관되게 표현되었는가.
2. **인물 포함**: 모든 장면에 실제 스토리 핵심 인물이 포함되었는가.
3. **오브젝트 포함**: 장면 내 핵심 오브젝트가 모두 반영되었는가.
4. **장소 포함**: 장면 내 핵심 장소가 모두 반영되었는가.
5. **관계성**: 등장인물과 오브젝트 간 관계가 명확히 표현되었는가.
6. **그림체 일관성**: 스토리 전반에 걸쳐 그림체가 일관되게 유지되었는가.

○ **평가 방식**:

1. 여러 스토리를 생성하고 각 스토리에 대해 Prompt Making 과정을 거쳐 프롬프트를 제작.
2. 생성된 프롬프트를 기반으로 각 항목 점수를 산출.

항목별 점수 및 평균을 통해 **Prompt Maker 성능 평가** 및 개선 방향 도출.

4.1.3. 초기 평가

○ 초기 평가에서 10 번의 생성에 대해 6.5/18 점 획득.

○ 등장인물 및 핵심 오브젝트에 대한 **정확한 묘사 반영 필요**.

○ 그림체 표현이 장면별로 상이하므로 **스타일 일관성 확보 필요**.

○ 등장인물과 오브젝트 간 **관계성**을 명확히 기술하여 시각적 의미를 보완.

4.1.4. Scene parsing, prompt making 개선 후 평가

○ 개선 후 평가에서 10 번의 생성에 대해 17/18 점 획득.

○ 오브젝트의 누락이 생기는 문제는 개선할 수 없었음.

○ 이는 향후 다른 버전의 Llm 모델로 실험 요망.

4.2. Image Making 평가

본 연구에서는 **Image Making 과정에서 생성된 이미지의 품질**을 정성적 요소를 기반으

로 정량적으로 평가하였다. 이를 통해 프롬프트 생성 과정의 개선점을 도출하고, 개선된 Image Maker의 성능을 검증하는 데 활용하였다.

4.2.1. 평가 목적

- ○ 이미지 제작 과정에서 **등장인물, 장소, 오브젝트, 관계성, 그림체 일관성** 등의 요소가 프롬프트의 지시대로 충분히 반영되었는지 평가.
- ○ 평가 결과 및 비용 등을 고려, 최종 서비스 제공 방식 결정 시 참고 지표로 활용 예정.

4.2.2. 평가 지표 및 방법

- ○ 수행 시간 측정
- ○ 각 프롬프트에 대해 Scene 단위로 이미지를 생성하고, **정성적 기준**을 0~3 점 척도로 정량화하여 평가.
- ○ 평가 항목:
    1. **등장인물 묘사 반영**: 모든 장면에서 핵심 인물 묘사가 이미지에 반영되었는가.
    2. **인물 포함**: 모든 장면에 프롬프트에서 지시된 핵심 인물이 포함되었는가.
    3. **오브젝트 포함**: 장면 내 핵심 오브젝트가 모두 반영되었는가.
    4. **장소 포함**: 장면 내 핵심 장소가 모두 반영되었는가.
    5. **관계성**: 등장인물과 오브젝트 간 관계가 반영되었는가.
    6. **그림체 일관성**: 스토리 전반에 걸쳐 그림체가 일관되게 유지되었는가.
- ○ **평가 방식**:
    1. 파이프라인 과정을 거치며 생성된 각 프롬프트에 대해 Image Making 과정을 거쳐 프롬프트를 제작.
    2. 생성된 이미지를 기반으로 각 항목 점수를 산출.
    3. 항목별 점수 및 평균을 통해 **Image Maker 성능 평가** 및 개선 방향 도출.

4.2.3. 초기 평가

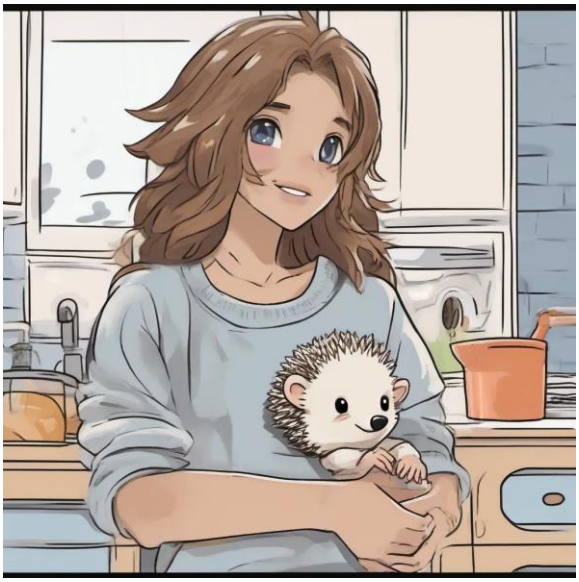
- ○ 초기 평가에서 10 번의 생성에 대해 6/18 점 획득.
- ○ 등장인물 및 핵심 오브젝트에 대한 **정확한 묘사 반영 필요**.
- ○ 의상, 인물 외형 묘사 반영 부족.
- ○ 이미지 생성 시간 4~5 분.

4.2.4. Image Making 개선 후 평가

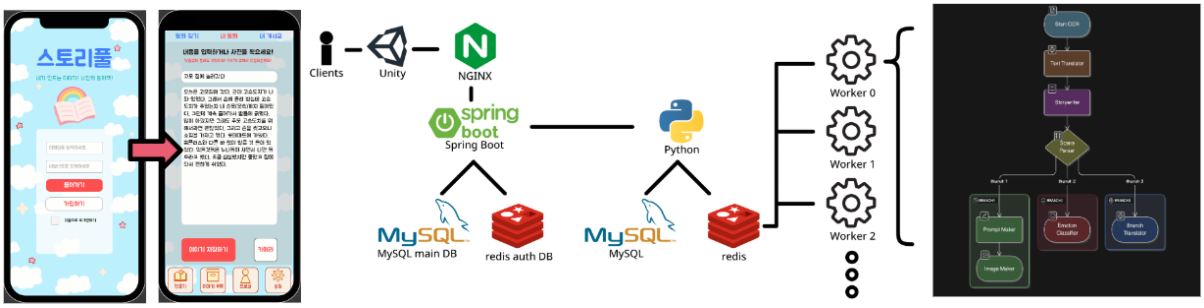
○ 개선 후 평가에서 10 번의 생성에 대해 8/18 점 획득.
○ 모델 변경, 프롬프트 길이에 따른 청크 분할과 multi embeding 방식 활용.
○ 프롬프트 길이에 따라 누락 요소 발생을 완벽히 통제할 수 없었음.
○ 향후 instance diffusion 과 같은 모델로 실험 요망.
○ 이미지 생성 시간 4~5 분.

4.2.5. 서버리스 Image Making 과의 비교

○ Hugging face 에서 제공하는 서버리스 inference api 사용, 동일 모델 사용.
○ Quantization 등으로 인해 같은 프롬프트더라도 프롬프트 반영 정확도 및 수행시간에서 차이 발생.
○ 특히, 수행 시간은 서버리스 추론 시 15 초 내외로 16 배 차이 발생.
○ 로컬 환경에서의 한계점이 명확이 보였음.

	
Local scene 1	Local scene 2
	
Hugging face serverless inference scene 1	Hugging face serverless inference scene 2

5. 결론 및 향후 연구 방향



본 프로젝트에서는 어린이가 작성한 글을 기반으로 AI 모델을 활용하여 자동으로 그림 동화를 생성하고 이를 공유할 수 있는 플랫폼을 만들었다.

- ○ OCR 을 활용하거나 직접 사용자로부터 내용을 입력받는다.
- ○ 서버 기반 AI 모델과 연계하여 동화에 필요한 내용을 생성한다.
- ○ 작성된 동화를 플랫폼을 통해 공유하고 타인의 동화를 평가할 수 있다.
- ○ 실사용자에 맞는 어린이 친화적 인터페이스를 제공한다.

이를 통해 사용자는 손쉽게 자신의 글을 동화로 변환하여 시각적/청각적 경험을 동시에 즐길 수 있다. 또한, 클라이언트-서버-AI 모델 연동을 통해 실시간으로 동화를 생성하고 이를 공유하는 플랫폼을 통해 많은 사람과 감정적인 교류를 할 수 있고 피드백을 받는 것이 가능하다.

하지만 생성된 동화의 결과가 사용자가 입력한 내용을 완벽하게 반영하지는 못하였다. LLM모델을 사용할 때 양자화를 진행하였는데, 이때 하드웨어의 성능이 뒷받침해주지 못해 결과에 영향을 주었다. 이미지 생성 모델에서도 사용한 모델이 오픈소스 모델이라 높은 성능을 보이지는 못하였다.

본 프로젝트에서는 클라우드 환경의 Auto Scaling 기능을 직접 적용하지는 않았다. 초기 개발 단계에서는 사용자 트래픽이 제한적일 것으로 예상되었고, 핵심 기능 검증(MVP)에 집중하는 것이 우선이라 판단했기 때문이다. 다만, 실제 서비스 확장 시에는 동시 접속자 처리 및 서버 자원 최적화를 위해 Auto Scaling이 필요하며, 이를 직접 실험하지 못한 점은 아쉬움으로 남는다.

또한, 대규모 비동기 작업 분산을 위해 AWS SQS(Simple Queue Service) 같은 메시지 큐 시스템을 도입할 수 있었으나, 본 프로젝트의 작업량과 규모를 고려했을 때 필수적이지

않다고 판단하였다. 설계 복잡성과 운영 부담을 줄이고자 단순한 구조를 유지했지만, 결과적으로 다중 작업 실행 시 처리 효율성을 확보하지 못한 아쉬움이 있다. 향후 서비스 고도화 단계에서는 메시지 큐를 통한 작업 분산이 반드시 고려되어야 한다.

동화를 사용자에게 제공할 때 시각효과를 넣기 위해 동화에 나오는 주요 객체에 대한 분리 후, 각 객체별로 애니메이션을 제공하고자 하였다. 하지만, 객체별 분리를 수행하려면 생성에 있어 시간이 훨씬 더 많이 소요되었기 때문에 이를 기반으로 한 서비스를 제공할 수는 없었다.

본 프로젝트의 한계를 보완하기 위해, 향후 연구는 다음과 같이 진행될 수 있다.

1) LLM 및 이미지 생성 모델의 성능을 개선하고 하드웨어 최적화를 통해 동화 생성 정확도와 시각적 품질을 높이는 연구

2) 클라우드 환경에서 Auto Scaling과 메시지 큐 기반의 비동기 작업 분산을 도입하여 다중 사용자 동시 처리와 서비스 확장성 확보

3) 동화 내 주요 객체 분리와 객체별 애니메이션 제공을 연구하여 몰입형 시각 효과를 구현 및 최적화 기술 탐구

4) 사용자 맞춤형 인터페이스와 멀티모달 입력을 통합하여 어린이 친화적이고 상호작용적인 플랫폼으로 발전시키는 방향 요구

6. 참고 문헌

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models," *arXiv preprint arXiv:2307.01952*, 2023.

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *arXiv preprint arXiv:1910.03771v2*, 2019.

[3] Nils Reimers and Iryna Gurevych, "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks," *arXiv preprint arXiv:2108.08877*, 2021.

[4] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, et al., "LLaMA: Open and Efficient Foundation Language Models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, et al., "Attention is All You Need," *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 5998-6008, 2017.

[6] Banghao Chen, Yuanzhe Chen, Xue Jiang, et al., "Unleashing the Potential of Prompt Engineering in Large Language Models: A Comprehensive Review," *arXiv preprint arXiv:2310.14735*, 2023.

[7] The University of Western Australia, "Emotional Psychology: Understanding Emotions and Feelings," University of Western Australia, 2023. [Online]. Available: https://online.uwa.edu/news/emotional-psychology