

허재성's Portfolio

+82 010-5209-5605
cork2586@naver.com

<https://github.com/jwelyl>

About Me



🎓 EXPERIENCE

- Tmax GAIA UX 연구원 (2023.08.16 ~)
- SSAFY 팀 프로젝트 (2023.01 ~ 2023.06)
- SSAFY 웹 개발 과정 (2022.07 ~ 2022.11)
- 서강대학교 컴퓨터공학부 (2016.02 ~ 2022.08)
- 한가람고등학교 (2011.02 ~ 2014.02)

🏆 Awards /Certificates

- 2023.06.16 SSAFY Certificate 등급 우수 (상위 30%)
- 2023.05.30 SSAFY Android 프로젝트 환상일기 결선 우수상
- 2023.05.26 SSAFY Android 프로젝트 환상일기 우수상
- 2023.02.17 SSAFY Web 기술 프로젝트 Yuj 우수상

About Me



주요 기술

실제 프로젝트에서 주로 사용한 기술들입니다.



Java

- 객체지향적으로 코드를 작성할 수 있습니다.
- 상속, 다형성을 이해하고 활용할 수 있습니다.
- 레거시 코드를 읽고 활용할 수 있습니다.



Spring

- DI, IoC, Bean 등 Spring 핵심 개념을 숙지하고 활용이 가능합니다.
- Spring MVC를 이해하여 Domain, Controller, Service를 구분하여 설계, 구현한 경험이 있습니다.
- JPA를 적용하여 Data Access를 구현할 수 있습니다.
- File Upload/Download가 가능한 Back-End Application을 구현한 경험이 있습니다.



MySQL

- 도메인 요구사항에 맞게 ER-Diagram을 설계하고 관계형 DB relation으로 구현할 수 있습니다.
- DDL과 JOIN을 활용한 복잡한 쿼리를 작성할 수 있습니다.
- 필요에 따라 정규화, 역정규화를 적절히 선택하여 테이블을 설계한 경험이 있습니다.

About Me



최근 관심있는 기술

개인적인 학습, 실무에서 접하면서 관심을 가지게 된 기술들입니다.



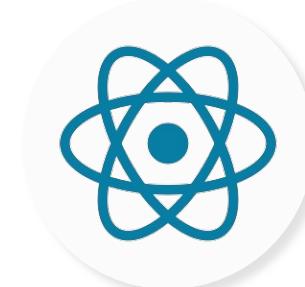
Kotlin

- 여러 분야에서 Java의 대체 언어로 각광 받고 있어 학습하는 중입니다.
- Null Safety를 통해 컴파일 시점에서 오류를 감지하는 코드를 작성해보았습니다.
- Java로 해결 가능한 프로그래밍 문제를 Kotlin으로도 해결할 수 있습니다.



TypeScript

- TypeScript를 이용해 객체지향적인 코드를 작성할 수 있습니다.
- ES6 문법을 활용하여 작성된 레거시 코드를 분석하고 활용하는 실무를 하는 중입니다.



React

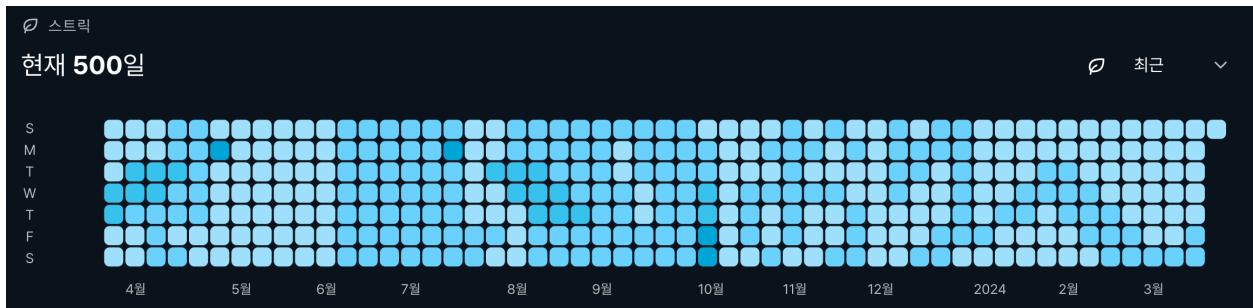
- useState, useEffect hook을 활용할 수 있습니다.
- props를 이용해 Component 간의 통신을 구현할 수 있습니다.
- 레거시 함수형 Component를 활용할 수 있습니다.

About Me



꾸준함을 즐기는 개발자

- 매일 한 문제씩 알고리즘 문제 풀기를 좋아합니다.
- 어제는 풀지 못한 문제를 오늘은 풀수 있는 것을 기뻐합니다.
- 다양한 알고리즘을 학습하며 더 나은 개발자가 되어가는 중입니다.



Problem Solving (Algorithm & SQL)

BOJ 1414번 불우이웃돕기

다음은 불우이웃돕기 기금 활용률 하락 위기 부족을 풀지 못하였다. 아직 일정을 맞춰서 모든 행정이 되어온 기관이나 단체가 있다. 이들 단체는 이미 기금이 필요 없고 그로 인해 다른 단체들이 행정을

<https://www.acmicpc.net/problem/1414>

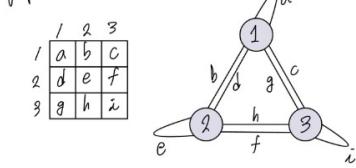
BAEKJOON
ONLINE JUDGE

1414번 불우이웃돕기
문제의 정점을 가진 그래프가 주된 항목 형태로 주어진다. 이를 사용하여 간선 비용이 문자로 주어진 것, 그리고 자

기 자신으로 가는 단선도 주어졌으며 단방향 그래프라는 점은媒介란 graph[v1][v2]가 graph[v2][v1]이 둘 다

기여하는 경향이 있다는 것이다.

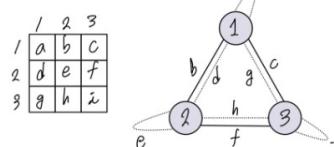
예제 1을 그림으로 나타내면 다음과 같다.



문제의 조건에 의하면 모든 정점이 이어지도록 최소 크기 간선만 남기고 나머지는 기부하는 것이다. 즉 최소 비용

최단 트리를 만들고, 원래 그때의 간선에서 MST에 포함되지 않는 간선들은 기부해야 되는 것이다.

graph



최소 비용 신장 트리를 만드는데 고려할 필요가 있는 간선들은 정점과 연결되는 간선은

최소 비용 신장 트리를 만드는데 현재 도움이 되지 않는다. 또한 두 정점 사이에 간선이 여러 개일 경우, 가중치가

작은 간선만 고려해도 된다. 위의 그림에서 실선으로 된 간선이 바로 간선이다. 후보 간선은 거리로 Kruskal

Algorithm을 적용하여 mst의 비용을 구하고 현재 간선 비용에서 제외한다.

그대로 구축 시 주의 사항으로 graph[v1][v2] == 0이면 v1 ~ v2 사이 간선이 없다는 뜻이다. 예외 처리를 해서

간선을 추가하지 않도록 Integer.MAX_VALUE 값을 주면, 으로 계산하는 실수를 쓰면서 안된다.

Problem Solving (Algorithm & SQL)

BOJ 19641 중복 집합 모델

19641번 중복 집합 모델

SQLOL 문제입니다. 나중에 수 있는 데이터를 나타내야하는 중복한 표현법을 찾고 있지만,

제공된 구조를 표현하기가 한계가 있다. 이런 한계를 보완해보자 나중 모임 중 하나나

<https://www.acmicpc.net/problem/19641>

BAEKJOON
ONLINE JUDGE

Problem Solving (Algorithm & SQL)

BOJ 12014 주식

12014번 주식

어느 날 당신은 출근길에, 저녁길에 산책길에서 놀라운 문제를 발견해졌다. 이 문제는 미

래피 페스티벌 주제가 되면 반드시 푸시고 싶다. 절대 하는 마음으로 미화되거나 거기

<https://www.acmicpc.net/problem/12014>

BAEKJOON
ONLINE JUDGE

LIS의 문제를 구하는 문제였다. 테스트케이스 개수 T가 최대 100이고, N이 최대 10^5 으로 DP를 이용한

이거나 O(NlogN) 풀이로 어떤 시간 초과가 날 것이다. 따라서 이분 탐색을 이용한 O(NlogN) 풀이를 고민해야 한다.

1	2	3	4	5	6	7	8	9	10	...
100	50	70	90	75	87	105	78	110	60	

LIS

21 : -00

1 : -00 100 100 > -00 ④, 1이 = 1

2 : -00 50 50 < 100 ④, 1이 = 1

3 : -00 70 70 < 50 ④, 1이 = 2

4 : -00 90 90 < 70 ④, 1이 = 3

5 : -00 75 75 < 90 ④, 1이 = 3

6 : -00 87 87 < 75 ④, 1이 = 4

7 : -00 105 105 < 87 ④, 1이 = 5

8 : -00 105 105 < 105 ④, 1이 = 5

9 : -00 110 110 < 105 ④, 1이 = 6

10 : -00 110 110 < 110 ④, 1이 = 6

초기 LIS 리스트에 더미 값 Int.MIN_VALUE를 넣고 시작한다. LIS에는 수가 오름차순으로 저장된다.

주어진 수들의 수를 미리 확인한다.

1. LIS에 저장된 마지막 값보다 큼 경우

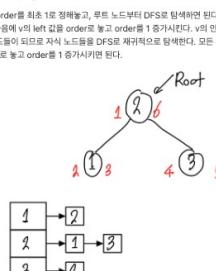
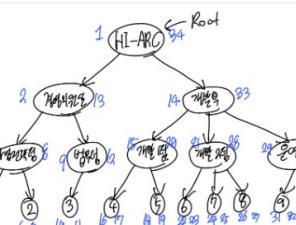
LIS에 추가한다. 이 경우 LIS의 길이가 1증가한다.

2. LIS에 저장된 마지막 값보다 작을 경우

LIS에 저장된 수 중 해당 수보다 크거나 같은 최소 수를 찾는다. 이 때 이분탐색을 이용해 O(logN)에 찾을 수

있다. 찾은 수를 현재 수열의 수로 대체한다.

수열의 모든 수를 확인한 후, 이미 길을 제외한 LIS의 길이 - 1이 정답이다.



부트가 정해진 root로부터 DFS를 하면 된다. 인접 리스트로 트리를 구현한 템플릿(정점 개수가 최대 10^5 개이

므로 그래야만 한다.) 각 정점별 인접 리스트를 오름차순으로 정렬하여 자식 정점 중 번호가 작은 정점부터 탐색하

도록 한다.

부트가 정해진 root로부터 DFS를 하면 된다. 인접 리스트로 트리를 구현한 템플릿(정점 개수가 최대 10^5 개이

므로 그래야만 한다.) 각 정점별 인접 리스트를 오름차순으로 정렬하여 자식 정점 중 번호가 작은 정점부터 탐색하

도록 한다.

<https://solved.ac/profile/koreii>

About Me

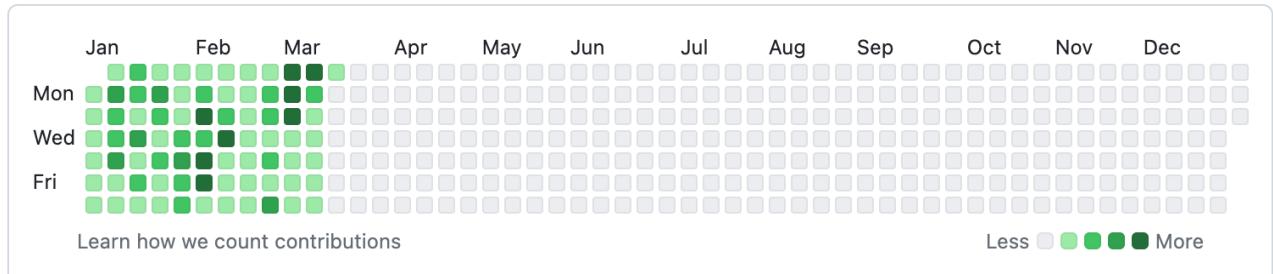


어제보다 더 똑똑해진 나

- 2024년부터 데일리 인증 스터디를 통해 매일 학습한 내용을 GitHub에 comit하고 있습니다.
- 1일 1 알고리즘 학습 뿐만 아니라, Computer Science, Spring, JPA 등 다양한 학습 내용들을 정리하여 올리고 있습니다.
- 컴퓨터 과학 학습이 공부가 아니라 취미가 되기를 지향합니다.

410 contributions in 2024

Contribution settings ▾



2024.01. 데일리 인증

2024.01.mon ~ 2024.01.06.sat 데일리 인증

2024.01.01

BOJ 1025 (Bruteforcing Algorithm)

2024.01.02

Git Branch, Merge & Conflict, Rebase vs Merge, Computer Network TCP vs UDP, Queueing Delay, BOJ 20010 (MST, DFS)

2024.01.03

Computer Network TCP/IP 5 Layers, Application Layer, HTTP, Transport Layer, BOJ 12763 (Dijkstra Algorithm, DFS)

2024.01.04

Computer Network Socket, Socket API, BOJ 2250 (Tree, DFSm Preorder, Inorder, Postorder)

2024.01.05

Computer Network TCP/UDP Demultiplexing, UDP Header, BOJ 20530 (DFS, Tree, Cycle)

2024.01.06

BOJ 14570 (Tree, DFS)

2024.01.07.sun ~ 2024.01.13.sat 데일리 인증

2024.01.07

BOJ 12896 (Tree, DFS)

2024.01.08

JPA Project Setting, Entity Manager Factory, Entity Manager, Transaction, Computer Network Go-Back-N, Selective Repeat, BOJ 13016 (Tree, DFS)

2024.01.09

Spring Container Bean Registration, Bean Definition, Computer Network TCP, TCP Segment Header, BOJ 18251 (Tree, DFS, Kadane Algorithm)

2024.01.10

JPA Persistence Context, BOJ 16947 (DFS, BFS, Cycle)

2024.02. 데일리 인증

2024.02.01.thu ~ 2024.02.03.sat 데일리 인증

2024.02.01

JPA 연관관계 매핑, Computer Network DHCP, BOJ 11509 (Greedy Algorithm)

2024.02.02

Computer Network Routing vs Forwarding, Link State Algorithm, BOJ 4792 (MST)

2024.02.03

JPA 연관관계 매핑 (일대다 매핑), BOJ 1798 (Geometry)

2024.02.04.sun ~ 2024.02.10.sat 데일리 인증

2024.02.04

BOJ 13202 (Geometry)

2024.02.05

JPA 연관관계 매핑 (일대일 매핑, 디데디 매핑) Computer Network Distance Vector Algorithm, BOJ 14926 (DFS, Greedy Algorithm)

2024.02.06

JPA 상속관계 매핑, BOJ 7453 (Meet in the middle), Java Arrays.sort() vs Collections.sort(), SQL Join

2024.02.07

JPA Proxy, BOJ 12014 (Longest Increasing Subsequence)

2024.02.08

JPA Eager Loading vs Lazy Loading, Computer Network Hierarchical Routing & Autonomous System, BOJ 13114 (Two Pointer)

2024.02.09

JPA Data Type, BOJ 6581 (String, Parsing)

2024.02.10

BOJ 2773 (Geometry, Vector)

2024.02.11.sun ~ 2024.02.17.sat 데일리 인증

https://github.com/jwelyl/daily_certification

주요 팀 프로젝트

羽毛 환상 일기

- SSAFY 8기 최종 프로젝트 결선 우수상 (148개 팀 중 상위 8위)
- SSAFY 8기 최종 프로젝트 최우수상 (서울 1반 8개 팀 중 1위)
- REDAME

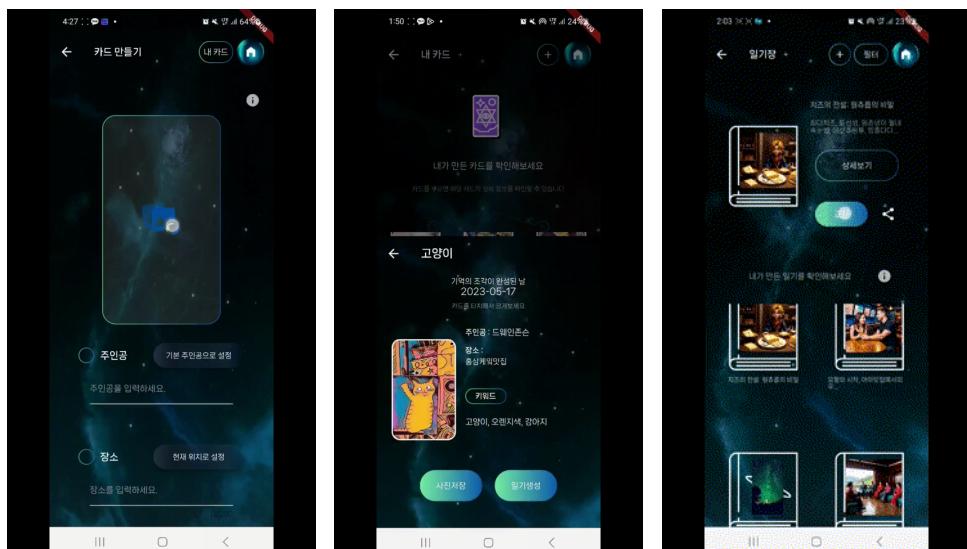
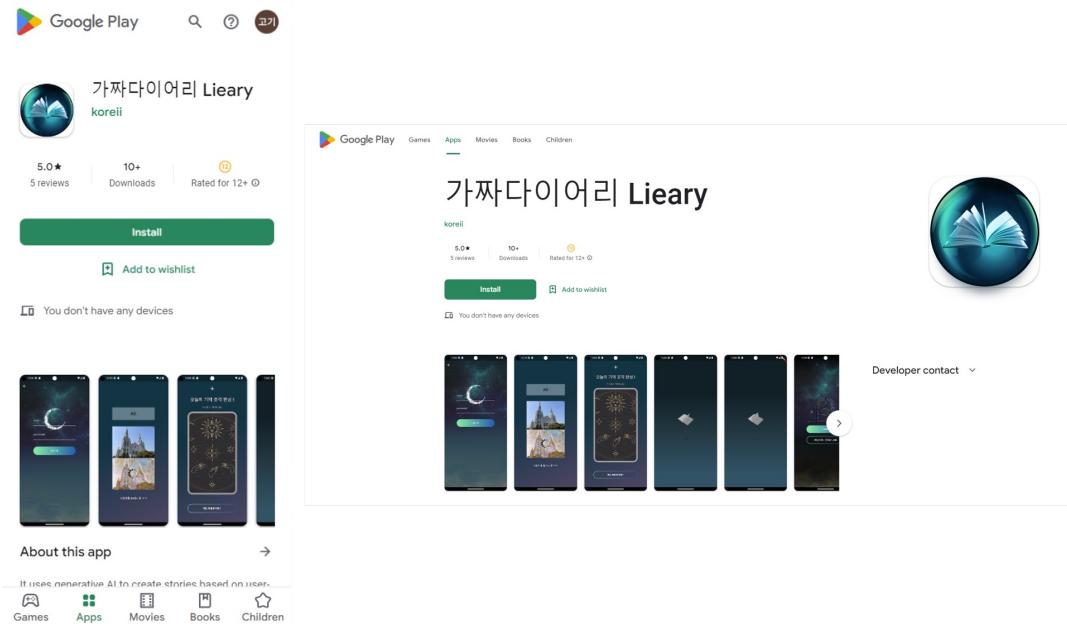
<https://github.com/jwelyl/fakediary/blob/develop/README.md>

Project

(feather) 환상 일기

- ChatGPT를 활용하여 하루동안 있었던 일을 재미있는 일기로 작성해 주는 Android 앱입니다.
- ChatGPT, Stable Diffusion API를 호출하여 다이어리로 생성해줍니다.

<https://github.com/jwelyl/fakediary/blob/develop/README.md>



Project



환상 일기

❖ 기간

- 개발 기간 : 2023.04.10 ~ 2023.05.19 (6주)
- 결선 발표 기간 : 2023.05.22 ~ 2023.05.30 (1주)

❖ 참여 인원

- Back-End 개발 : 허재성(본인) 포함 3명
- Android 개발 : 3명

❖ 나의 역할

- 팀원 (Back-End)
- 관계형 데이터베이스 설계 및 구현
- DeepArtEffects를 활용하여 디바이스에서 촬영한 사진을 명화풍의 카드 이미지로 변환
- ChatGPT를 이용하여 키워드를 이용한 일기 컨텐츠 생성
- Stable Diffusion을 이용하여 일기의 삽화 생성
- SOUNDRAW를 크롤링하여 배경 음악 다운로드 및 일기에 삽입
- Spring Batch를 이용하여 SOUNDRAW에서 배경 음악 일괄 크롤링 및 다운로드
- 일기 랜덤 교환 구현

❖ 성과

SSAFY 8기 최종 프로젝트 결선 우수상 (148개 팀 중 상위 8위)

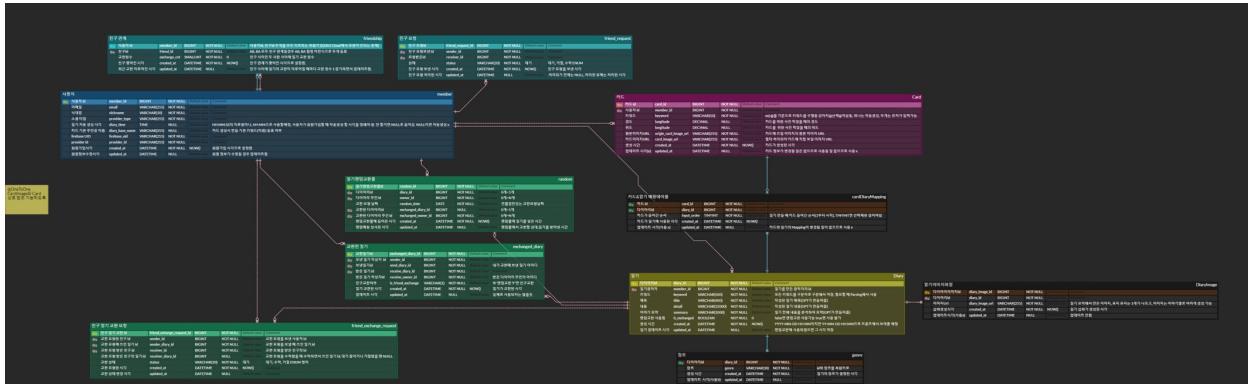
SSAFY 8기 최종 프로젝트 최우수상 (서울 1반 8개 팀 중 1위)

Project



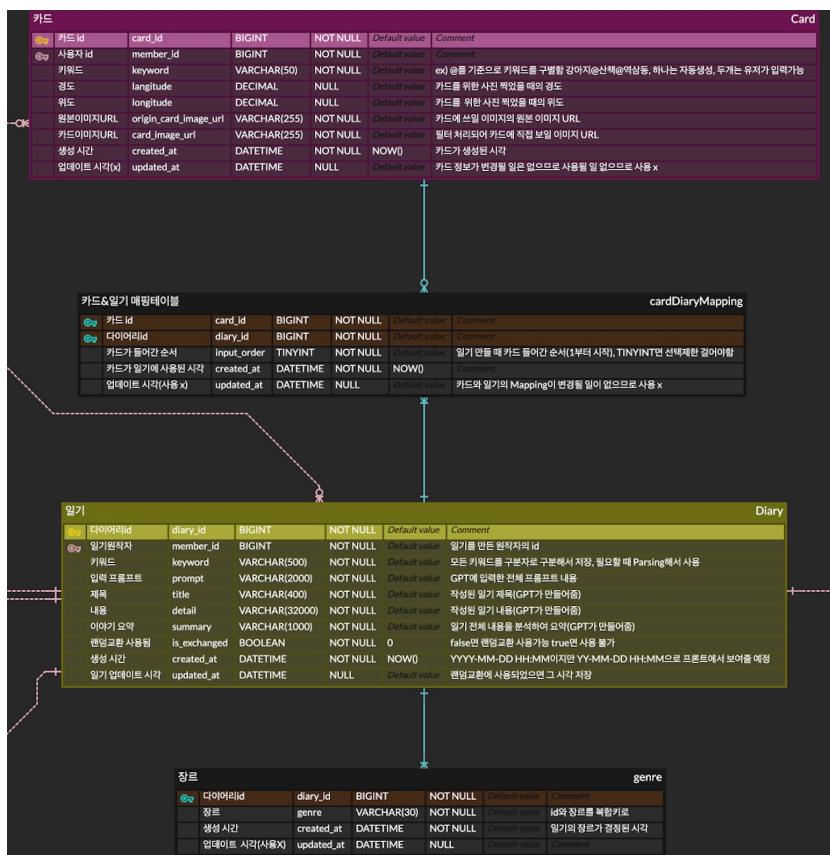
‘환상 일기’에서의 나의 역할

전체적인 관계형 데이터베이스 설계 및 구현



<https://www.erdcloud.com/d/XsCovTydibYELpEaH>

핵심 테이블인 카드, 일기 테이블 설계 및 구현



Project



‘환상 일기’에서의 나의 역할

관계형 데이터베이스 설계 및 구현

어려웠던 점

1. 카드 테이블 정의

개요

사용자가 Android 기기로 사진을 촬영하면 해당 사진을 기반으로 카드가 생성됩니다. 사용자는 카드에 최대 3개의 키워드를 입력할 수 있습니다. 카드에 저장된 키워드를 기반으로 ChatGPT가 일기를 생성해주므로 매우 중요한 정보였습니다.

카드에 저장될 수 있는 키워드를 어떻게 데이터베이스에 저장할지 정하는 것이 어려웠습니다. 다음과 같은 방안을 고려해보았습니다.

키워드 개수의 제한 x

최초에는 카드에 저장될 키워드의 개수를 제한하지 않기로 했습니다. 하지만 키워드의 개수가 너무 많아질 경우, ChatGPT가 키워드를 기반으로 일기를 생성할 때 일기의 퀄리티가 단순 키워드의 나열이 되는 문제가 있었습니다. 또한 하나의 카드가 여러 개의 키워드를 가지게 하기 위해서는 **카드-키워드의 1:N 관계를 가지는 키워드 테이블을 생성해야** 했습니다. 키워드 테이블을 분리할 경우 카드 테이블 내에 키워드를 따로 저장할 필요는 없지만 대신 카드로부터 키워드 정보를 찾기 위해서는 JOIN을 통해 찾아야 하는 문제가 있었습니다.

키워드의 개수를 3개로 제한 후 3개의 키워드 각각을 column화

카드에 저장될 키워드 개수를 최대 3개로 저장하고 각각을 카드 테이블에 키워드1, 키워드2, 키워드3으로 저장하는 방법을 생각했습니다. 따로 테이블을 생성해서 JOIN을 할 필요도 없고, 구현의 난이도도 어렵지 않다는 장점이 있었습니다. 하지만 키워드가 3개 미만이 들어올 경우, NULL 값이 채워지는 문제와, 후에 ‘환상 일기’ 서비스가 개선되어 더 많은 키워드를 저장할 수 있게 될 경우, 카드 테이블 구조를 변경해야 하는 문제가 있었습니다.

JOIN을 하지 않으면서도 후에 카드 테이블의 구조를 변경하지 않아도 되는 테이블 구조를 설계해야 했습니다.

Project



‘환상 일기’에서의 나의 역할

관계형 데이터베이스 설계 및 구현

어려웠던 점

1. 카드 테이블 정의

해결 방법

카드						Card
필드	카드 id	card_id	BIGINT	NOT NULL	Default value	Comment
사용자 id	member_id	BIGINT	NOT NULL	Default value	Comment	
키워드	keyword	VARCHAR(50)	NOT NULL	Default value	ex) @를 기준으로 키워드를 구별할 강아지@산책@역삼동, 하나는 자동생성, 두개는 유저가 입력 가능	
경도	longitude	DECIMAL	NULL	Default value	카드를 위한 사진 찍었을 때의 경도	
위도	latitude	DECIMAL	NULL	Default value	카드를 위한 사진 찍었을 때의 위도	
원본이미지URL	origin_card_image_url	VARCHAR(255)	NOT NULL	Default value	카드에 쓰일 이미지의 원본 이미지 URL	
카드이미지URL	card_image_url	VARCHAR(255)	NOT NULL	Default value	필터 처리되어 카드에 직접 보일 이미지 URL	
생성 시간	created_at	DATETIME	NOT NULL	NOW()	카드가 생성된 시각	
업데이트 시간(x)	updated_at	DATETIME	NULL	Default value	카드 정보가 변경될 일은 없으므로 사용될 일 없으므로 사용 x	

카드 테이블에 키워드 column을 유지하고, 대신 하나의 column을 받는게 아니라 Android 앱에서 받은 키워드들을 모두 합쳐서 단일 문자열로 만들어 저장하도록 구현했습니다. 키워드가 k1, k2, k3로 저장될 경우 @를 delimiter로 “k1@k2@k3”로 합쳐서 이를 카드 테이블의 키워드 column에 저장했습니다. 후에 일기를 생성하기 위해 카드를 조회할 경우, 넘겨줄 때는 @를 기준으로 다시 파싱하여 키워드를 복구해내는 방식으로 구현했습니다.

현실적인 키워드 개수를 10개 이하라고 가정했을 때 파싱에 오버헤드가 크지 않을 것이라 판단했고, 데이터베이스 테이블 구조도 단순화할 수 있었습니다.

아쉬웠던 점

만약 사용자가 입력한 키워드에 @가 포함될 경우 파싱 과정에서 문제가 발생하는데 이에 대한 대비책이 없었습니다. 최대한 사용자가 입력하지 않을 법한 문자를 delimiter로 사용하려는 노력을 했지만 근본적인 방지책은 아니었습니다.

MySQL에서는 JSON 형태로, Oracle DB에서는 VARRAY로 가변 배열을 저장할 수 있는데 프로젝트 진행 당시에는 사용해보지 못했습니다. 프로젝트를 리팩토링한다면 MySQL의 JSON을 활용해 delimiter로 파싱하는 임시방편이 아니라 가변 배열로 모든 키워드를 정확히 저장할 수 있도록 구현하겠습니다.

Project



‘환상 일기’에서의 나의 역할

관계형 데이터베이스 설계 및 구현

어려웠던 점

2. 일기 테이블 정의

개요

사용자가 만든 카드 중 1개 이상의 카드를 이용해 일기를 생성합니다. 카드에 저장된 키워드들을 전부 조합하고, 추가적으로 사용자가 일기 생성 시에 입력한 장르를 조합해서 일기를 생성합니다. 장르 또한 일기 생성에 사용되는 중요한 정보이므로 데이터베이스의 일기 테이블에 저장해야 합니다.

일기 생성에 사용된 키워드들은 카드에 사용된 키워드들의 조합이므로 중복을 제외시킨 후 카드 테이블에 저장한 것과 같은 방법으로 저장했습니다. 하지만 장르를 일기 테이블에 어떻게 저장할지는 키워드와는 다른 부분을 고민해야 했습니다.

사용된 장르들을 합하여 저장

일기에 사용된 장르들이 g1, g2, g3라 할 때, 키워드를 저장하는 방법과 같이 “g1@g2@g3” 형태의 문자열로 합쳐서 저장하는 방법을 생각해봤습니다. 하지만 키워드는 사용자가 임의로 입력해서 어떤 값이 들어올지 예측이 불가능한 것과 다르게, 장르는 종류가 호러, 액션, 스릴 등 총 10종류로 앱에 미리 정해져 있었고, 새로운 종류가 추가된다고 해도 그 종류가 정해져 있을 예정이었습니다. 또한 어떤 장르의 일기가 많이 생성되었는지 추적하기 위해서 장르를 통해 일기를 검색할 수도 있어야 했습니다. 키워드와 같은 방식으로 일기 테이블의 한 컬럼에 장르들을, 그것도 @로 합쳐서 저장한다면 이는 매우 비효율적이었습니다.

일기 테이블에 각 장르별 column을 boolean 값으로 추가

다음으로 생각해본 방법은 일기 테이블에 각 장르별로 column을 추가해서 해당 장르가 포함되면 true, 포함되지 않는다면 false로 저장하게 한다면 쿼리를 통해 특정 장르들을 포함하는 일기를 검색하는 쿼리를 작성하는 것은 수월할 것입니다. 하지만 테이블의 column 개수나 10개나 증가하는 문제가 있었고, 만약 앱에서 장르의 종류가 추가, 변경되거나 삭제될 경우 일기 테이블이 변경되어야 하는 문제가 있었습니다.

Project

‘환상 일기’에서의 나의 역할

관계형 데이터베이스 설계 및 구현

어려웠던 점

2. 일기 테이블 정의

해결 방법

일기						
	다이어리id	diary_id	BIGINT	NOT NULL	Default value	Comment
일기작성자	member_id	BIGINT	NOT NULL	Default value	일기를 만든 회원자의 id	
키워드	keyword	VARCHAR(500)	NOT NULL	Default value	모든 키워드를 구분자로 구분해서 저장, 필요할 때 Parsing해서 사용	
입력 프롬프트	prompt	VARCHAR(2000)	NOT NULL	Default value	GPT4 입력한 전체 프롬프트 내용	
제목	title	VARCHAR(400)	NOT NULL	Default value	작성된 일기 제목(GPT7가 만들어줌)	
내용	detail	VARCHAR(32000)	NOT NULL	Default value	작성된 일기 내용(GPT7가 만들어줌)	
이야기 요약	summary	VARCHAR(1000)	NOT NULL	Default value	일기 전체 내용을 분석하여 요약(GPT7가 만들어줌)	
랜덤교환 사용여부	is_exchanged	BOOLEAN	NOT NULL	0	false면 랜덤교환 사용가능 true면 사용 불가	
생성 시간	created_at	DATETIME	NOT NULL	NOW()	YYYY-MM-DD HH:MM:SS(YYYY-MM-DD HH:MM으로 프론트에서 보여줄 예정)	
일기 업데이트 시간	updated_at	DATETIME	NULL	Default value	랜덤교환에 사용되었으면 그 시간 저장	

장르						
	다이어리id	diary_id	BIGINT	NOT NULL	Default value	Comment
장르	genre	VARCHAR(30)	NOT NULL	Default value	Id와 장르를 복합키로	
생성 시간	created_at	DATETIME	NOT NULL	Default value	일기의 장르가 결정된 시간	
업데이트 시간(사용X)	updated_at	DATETIME	NULL	Default value	Comment	

장르 테이블을 분리하고 장르명과 해당 장르를 가진 일기의 PK를 복합 PK로 가지도록 하였습니다. 일기의 장르를 알고 싶을 때 일기의 PK를 통해 일기에 사용된 장르들을 알 수 있고, 반대로 장르에 해당하는 일기를 알고 싶을 때, 장르명을 통해 검색해 해당하는 일기들을 검색할 수 있습니다.

아쉬웠던 점

만약 앱에서 장르가 새로 추가될 경우 당장은 해당 장르의 일기가 존재하지 않으므로 장르 테이블에는 추가된 장르가 반영되지 않는 문제가 있습니다. 또한 (일기 PK, 장르명)의 복합키로 구분이 가능하지만 장르 테이블 안에 같은 장르명이 중복되어 들어가는 문제가 있었습니다.

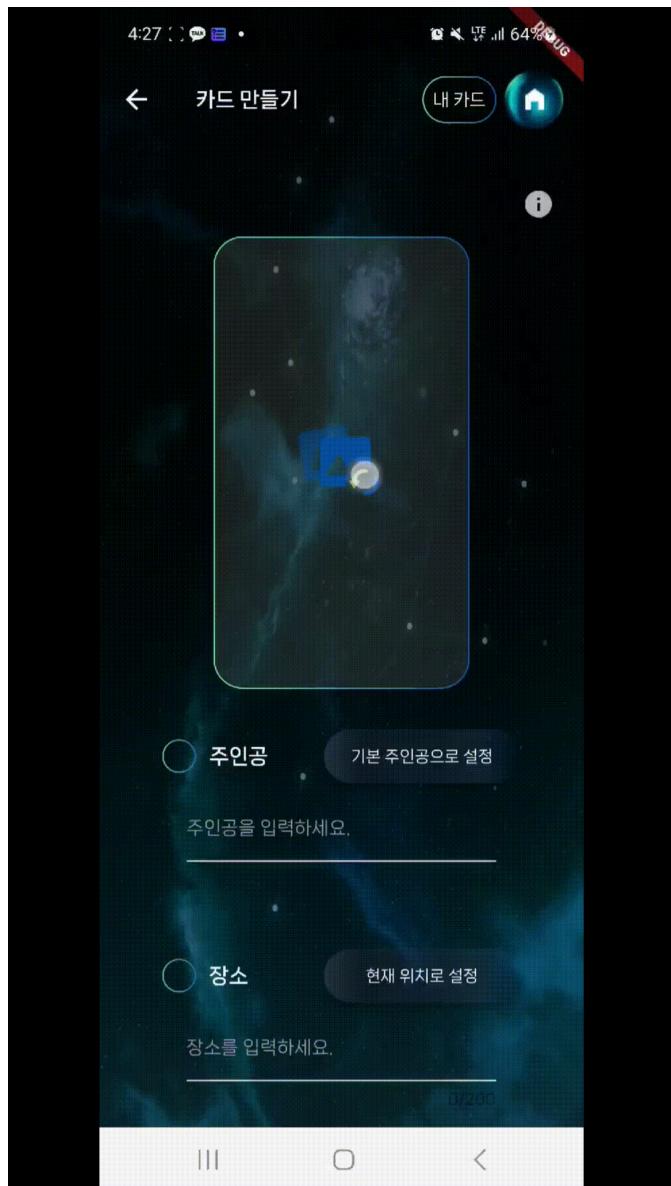
이러한 문제를 해결하기 위해 일기와 장르를 N:M 관계를 가지도록 테이블을 재설계해 중계 테이블 (ex) 일기_장르 테이블)을 만들고 일기 테이블과 중계 테이블을 1:N, 장르 테이블과 중계 테이블을 1:N 관계를 가지도록 구현하는 것을 고려해볼 수 있을 것 같습니다.

Project



‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경



Project



‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경

어려웠던 점

1. 카드 생성이 완료되기 전에 종료되는 문제

개요

사용자가 만든 카드 중 일부를 선택해서 일기를 생성하기 때문에 일기를 생성하기 위해서는 카드가 필수로 필요합니다. 카드를 생성하기 위해서는 Android 기기로 사진을 촬영하고 촬영한 사진과 키워드를 통해 카드가 생성됩니다.

카드 생성에서 시간이 가장 오래 걸리는 부분은 촬영한 사진을 명화풍의 이미지로 변경하는 것입니다. DeepArtEffects API를 사용해서 촬영한 사진을 get 요청으로 DeepArtEffects 서버에 전송하면 submissionId를 얻어오고, 이 submissionId를 get 요청으로 DeepArtEffects 서버에 전송하여 명화풍으로 변경된 사진의 URL을 얻어왔습니다.

Project

‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경

어려웠던 점

1. 카드 생성이 완료되기 전에 종료되는 문제

```
/*  
 * @param submissionId : 업로드 id  
 * @return : 업로드 결과로 생성된 카드 이미지 URL  
 */  
public String getCardImageUrl(String submissionId) throws Exception {  
    HttpHeaders headers = new HttpHeaders();  
    headers.set("x-api-key", API_KEY);  
    headers.set("x-api-access-key", ACCESS_KEY);  
    headers.set("x-api-secret-key", SECRET_KEY);  
  
    WebClient webClient = WebClient.builder()  
        .baseUrl(BASE_URL)  
        .defaultHeader(HttpHeaders.CONTENT_TYPE,  
            MediaType.APPLICATION_NEOBONSAEHEOFr -> header.addAll(headers))  
        .build();  
  
    Instant start = Instant.now();  
    Mono<DeepArtEffectsImageUrlResponse> response = webClient.get()  
        .uri(uriBuilder -> uriBuilder  
            .path("/result")  
            .queryParam("submissionId", submissionId)  
            .build())  
        .retrieve()  
        .bodyToMono(DeepArtEffectsImageUrlResponse.class);  
  
    Thread.sleep(5000); // 5초간 대기  
    DeepArtEffectsImageUrlResponse res = response.block();  
  
    Instant end = Instant.now();  
    log.info("status = " + res.getStatus());  
    log.info("url = " + res.getUrl());  
    log.info("Elapsed time: " + Duration.between(start, end).toMillis() + " ms");  
  
    return res.getUrl();  
}
```

DeepArtEffects에 API 요청을 보내기 위해 Spring WebFlux의 WebClient를 사용했습니다. WebClient로 submissionId를 포함해 GET 요청을 보냈는데 위의 코드에서 강제로 5초를 기다리게 한 후 종료할 경우, 카드 변환이 완료되기 전에 종료되어 변환된 카드의 URL을 얻을 수 없었습니다.

크기가 작은 이미지들은 변환이 빨리 이루어져서 5초의 대기 시간으로 충분했지만 충분히 큰 이미지로 테스트해보지 않아서 위와 같은 잘못된 코드를 작성하게 되었습니다.

Project



‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경

어려웠던 점

2. 카드 생성에 너무 오랜 시간을 기다리는 문제

```
Thread.sleep(10000); // 10초간 대기  
DeepArtEffectsImageUrlResponse res = response.block();
```

크기가 큰 이미지도 변환이 완료될 정도로 충분히 오랜 시간을 기다리도록 대기하는 시간을 10초로 증가시켜보았습니다. 10초로 변경 이후, 촬영한 모든 사진에 대해서 정상적으로 변환이 이루어졌습니다.

하지만 두 가지 문제점이 존재했습니다.

- 테스트에 사용된 사진보다 더 큰 사진을 변환할 경우 10초로도 부족할 수 있습니다. 이 경우 변환이 완료되지 않고 종료될 것입니다.
- 일관되게 10초를 기다리므로 기존에 5초 대기로도 충분히 변환이 완료되었던 이미지들도 10초가 지난 후에 변환이 완료된 URL을 얻을 수 있었습니다.

보다 근본적인 해결을 위해서는 Client(이 문제에서는 환상 일기의 SpringBoot 서버)에서 Server(DeepArtEffects)에 주기적으로 요청을 보내어 변환이 완료되는 것을 주기적으로 확인하고, 변환이 완료되면 종료하도록 Polling 방식을 구현해야 했습니다.

Project

‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경

어려웠던 점

해결 방법

```
/*
 *
 * @param submissionId : 업로드 id
 * @return : 업로드 결과로 생성된 카드 이미지 URL
 */
public String getCardImageUrl(String submissionId) throws Exception {
    HttpHeaders headers = new HttpHeaders();
    headers.set("x-api-key", API_KEY);
    headers.set("x-api-access-key", ACCESS_KEY);
    headers.set("x-api-secret-key", SECRET_KEY);

    WebClient webClient = WebClient.builder()
        .baseUrl(BASE_URL)
        .defaultHeader(HttpHeaders.CONTENT_TYPE,
        MediaType.APPLICATION_JSON_VALUE -> header.addAll(headers))
        .build();

    Instant start = Instant.now();

    DeepArtEffectsImageUrlResponse res = null;

    while(true) {
        Mono<DeepArtEffectsImageUrlResponse> response = webClient.get()
            .uri(uriBuilder -> uriBuilder
                .path("/result")
                .queryParam("submissionId", submissionId)
                .build())
            .retrieve()
            .bodyToMono(DeepArtEffectsImageUrlResponse.class);

        res = response.block();
        if(res.getStatus().equals("finished"))
            break;
        Thread.sleep(1000); // 1초 간격으로 확인
    }

    Instant end = Instant.now();
    log.info("status = " + res.getStatus());
    log.info("url = " + res.getUrl());
    log.info("Elapsed time: " + Duration.between(start, end).toMillis() + " ms");

    return res.getUrl();
}
```

1초 간격으로 서버에 요청을 보내면서 변환이 완료되었는지 확인하고 변환이 완료되면(response의 status가 finished일 경우) 종료하는 Polling 방식을 구현했습니다. 작은 이미지, 큰 이미지 가리지 않고 변환이 완료되면 종료되도록 구현할 수 있었습니다.

Project



‘환상 일기’에서의 나의 역할

사진을 명화풍의 카드 이미지로 변경

어려웠던 점

아쉬운 점

외부 서비스인 DeepArtEffects와 Spring Back-End 서버가 API 통신을 하기 위해 Spring WebFlux의 WebClient를 사용했지만, WebClient는 비동기-논블로킹 웹 클라이언트로 Reactive Programming 패러다임을 구현하는 모듈이었습니다.

이러한 특성을 잘 이해하지 못한 채 그저 API 통신을 위한 용도로만 사용한 것 같아 아쉬웠습니다.

Project



‘환상 일기’에서의 나의 역할

일기 컨텐츠 생성

어려웠던 점

1. 일기 생성이 오래 걸리는 문제

개요

‘환상일기’의 핵심 컨텐츠인 일기는 만들어진 카드로부터 키워드를 받아서, 카드로부터 가져온 키워드들과 사용자가 정한 장르들을 조합해 프롬프트를 생성 후 ChatGPT API로 전송합니다. ChatGPT가 일기의 내용을 2000자 정도의 3개의 장으로 나누어서 생성하고, 각 장의 내용을 요약한 부제(subtitle)를 생성하여 Spring Back-End 서버로 전송해줍니다.

전달받은 각 장의 부제(subtitle)를 StableDiffusion API로 전송하여 각 장에 어울리는 삽화를 생성합니다.

한편 사용자가 입력한 장르에 맞는 배경음악을 생성하기 위해 SOUNDRAW를 크롤링해 장르에 맞는 음악을 S3 서버에 다운로드하고 이를 일기에 삽입합니다.

위의 과정을 거쳐 일기의 내용, 삽화, 배경음악이 어우러진 일기를 생성합니다.

ChatGPT를 이용한 일기 내용 생성, StableDiffusion을 이용한 삽화 생성, SOUNDRAW를 이용한 배경음악 생성, 이 3가지가 모두 이루어져야 일기가 완성되었기 때문에 일기 생성이 매우 오래 걸렸습니다. GPT 4를 이용한 일기 내용 생성에 평균적으로 2분, Stable Diffusion으로 삽화를 생성하는 시간이 평균적으로 1분이었고 이 두 시간은 더 감소시키기 어려웠습니다.

추가적으로 SOUNDRAW를 웹 크롤링하여 장르에 맞는 음악을 다운로드했는데, 1곡의 평균 다운로드 시간이 2분 정도 걸렸습니다. 음악까지 다운로드 완료되어 일기가 완성되는데는 평균적으로 5분이 걸렸습니다.

또한 사용 가능한 SOUNDRAW 계정이 하나뿐이라 동시에 크롤링할 수 있는 곡이 1곡뿐이라, 일기 생성 요청이 5개가 동시에 올 경우 가장 마지막에 생성되는 일기는 음악 다운로드에만 10분을 추가적으로 기다려야 했습니다.

Project



‘환상 일기’에서의 나의 역할

일기 컨텐츠 생성

어려웠던 점

해결 방법

일기의 배경음악의 경우 전적으로 일기의 장르에만 영향을 받았습니다. 즉 사용자가 카드를 선택해 키워드를 ChatGPT API로 전달하거나, Stable Diffusion이 생성한 삽화에 영향을 받지 않고 장르별로 미리 준비해 둘 수 있는 것이었습니다.

하루 무료 다운로드가 가능한 음악 개수가 50개이고, ‘환상 일기’에서 제공하는 일기 장르가 10개이므로, Batch 처리로 이용자가 적을 새벽 시간에 장르별로 5개씩 음악을 SOUNDRAW에서 미리 크롤링하여 S3에 다운로드해두었습니다.

일기가 생성될 때 매번 크롤링하는 것이 아니라 일기 장르에 맞는 음악을 S3에서 랜덤으로 선택하여 URL을 통해 바로 삽입할 수 있도록 구현했습니다.

이를 통해 음악 크롤링 밑 다운로드에 걸리는 시간 2분을 절약하여 일기를 생성하는데 걸리는 최소 시간 5분을 3분 이하로 감소시킬 수 있었습니다.

아쉬운 점

한정된 예산으로 인한 제약, 배경 음악이 일기의 장르에 의해서만 결정됨을 이용한 해결방법이었지만 일기 생성 시점에 음악이 생성되는 것이 아니라 이미 생성된 음악 중에서 장르에 맞는 음악을 고르는 방식이라 아쉬웠습니다.

사용자가 많아진다면 이전에 들었던 음악을 다시 듣게 되는 일이 필연적으로 발생할 것이고 이는 사용자 경험에 좋지 않을 것입니다.

Spring Batch를 이용해 특정 시작에 정해진 작업을 일괄 처리하는 코드를 구현해봤는데, Batch 처리를 위해 Spring Batch를 꼭 사용해야 하는 지에 대한 고민이 부족했던 것 같습니다. SpringFramework에서 제공하는 @Scheduled 어노테이션으로도 충분히 해결 가능한 수준의 구현을 Spring Batch를 이용해서 구현한 것 같습니다.

Project Review

‘환상 일기’를 마치며

좋았던 점

- Generative AI, 그 중에서도 가장 이슈가 되었던 ChatGPT를 이용해서 재미있는 어플리케이션을 개발할 수 있었던 점이 좋았습니다.
- Web Application의 Backend만 개발을 해봤는데, Backend 개발이 Web Application에만 국한되지 않고 Mobile Application에도 적용 가능한 보편적인 개발이 될 수 있음을 느낄 수 있어 뜻 깊었습니다.
- ChatGPT, Stable Diffusion의 프롬프트를 연구하고 제공하여 멋진 컨텐츠를 생성할 수 있어서 정말 즐거웠습니다.
- 6주라는 길지 않은 시간, 한정된 예산임에도, 최적의 기술 스택을 선정하고 활용하여 소기에 목적한 기능을 모두 구현할 수 있어서 즐겁고 배운것이 많았던 프로젝트였습니다.

아쉬웠던 점

- ChatGPT, Stable Diffusion, SOUNDRAW를 API 호출을 통해 단순하게 결과물만 이용한 것 같아 아쉽습니다.
- WebFlux의 원리를 더 깊게 학습해서 반응형 프로그래밍을 통해 비동기적인 코드를 개발하지 못하고, 단순히 외부 API 호출용으로 사용한 것 같아 아쉽습니다.
- Spring Batch의 원리와 활용처를 더 깊게 이해하지 못하고 일괄 처리에만 집중해서 사용한 것 같아 아쉽습니다.
- MySQL의 JSON으로 데이터베이스의 column에 배열 데이터를 저장하지 않고, 임시방편으로 문자열을 구분자로 구분하여 저장한 것 같아 아쉽습니다.

개선할 점

- ‘환상 일기’에 사용되었던 Generative AI들에 대해 더 깊이 학습한다면 더 뜻 깊은 프로젝트로 남을 것 같습니다.
- WebFlux를 학습하여 성능 개선을 도모할 부분이 있는지 확인해보겠습니다.
- Spring Batch를 더 깊이 학습하고 ‘환상 일기’ 프로젝트의 적재적소에 적용시켜보겠습니다.
- 사용하는 RDBMS에서 제공하는 스펙을 더 조사해보고 데이터베이스 설계를 개선시키겠습니다.