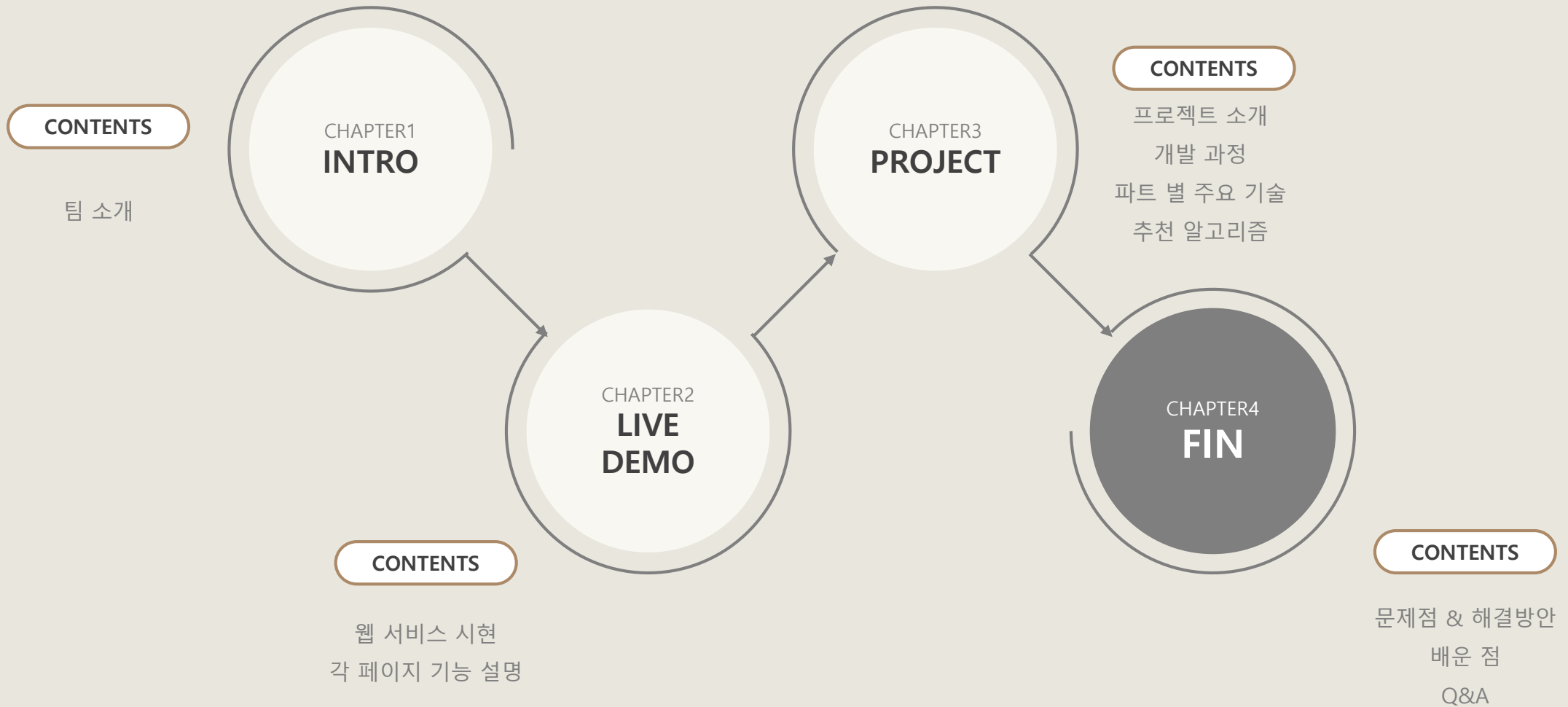


OTT 서비스 데이터 분석 프로젝트

TEAM 『BLACK PIZZA』

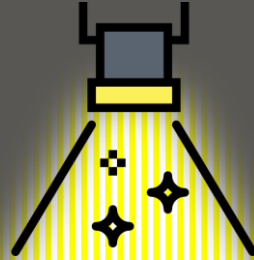
INDEX





BLACK PIZZA

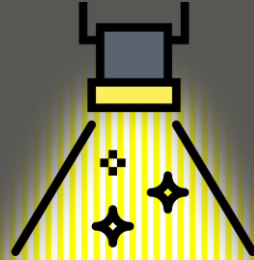
INTRO 팀 소개



DB

이재근
(팀장)

INTRO 팀 소개



DB

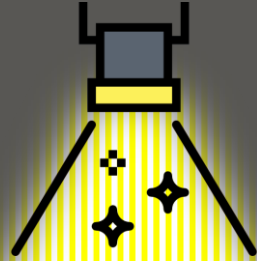
전현호



DB

이재근
(팀장)

INTRO 팀 소개



BE

이무원



DB

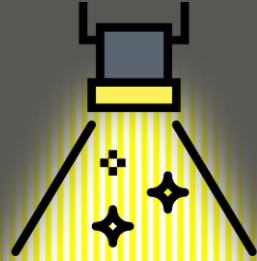
이재근
(팀장)



DB

전현호

INTRO 팀 소개



BE

장서우



DB

이재근
(팀장)



DB

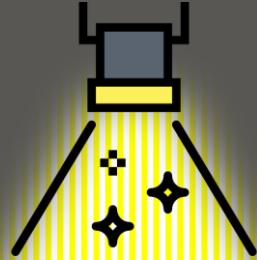
전현호



BE

이무원

INTRO 팀 소개



FE

김현하



DB

이재근
(팀장)



DB

전현호



BE

이무원



BE

장서우



PROJECT 프로젝트 소개



COVID-19 장기화

좀처럼 줄지 않는 코로나19 확진자
수로 인해 일상이 되어버린
비대면 시대



OTT 서비스의 성장

영화관 관객 수가 줄어들면서
집에서도 쉽게 미디어 콘텐츠를
즐길 수 있는 OTT 서비스의 이용량 증가

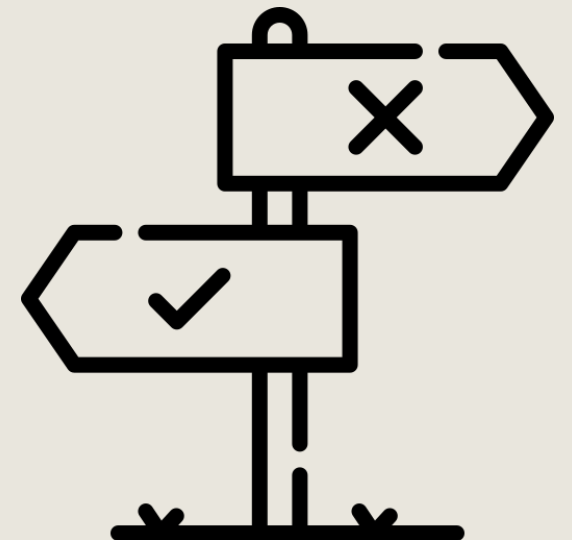


다양한 OTT 플랫폼

여러 플랫폼의 등장과 함께 플랫폼마다
이용할 수 있는 콘텐츠의 차이가 발생
유료 서비스인 만큼 소비자들은 합리적
소비를 위한 선택의 고민에 빠짐

다양해진 OTT 플랫폼 사이에서 나에게 딱 맞는 플랫폼은 어떻게 찾을까?

나만의 맞춤 콘텐츠와 OTT를 추천해주는 서비스를 만들어보자!





🎯 주요 타겟층

OTT 서비스를 사용하지 않는 미이용자



Why?

OTT를 처음 사용하는 사람들은 자신이 보고싶은 콘텐츠가 어느 플랫폼에 있는지 쉽게 확인하기 어렵다. 나에게 맞는 플랫폼을 추천해줌으로써 플랫폼을 찾는데 소요되는 **시간을 절약해주고** 여러 플랫폼에 동시에 구독 결제하는 일을 방지하여 **비용을 절감해준다.**



그라운드 룰

GitLab 사용 규칙(commit 메시지, MR assign)
및 프로젝트 기간 동안 팀원 간에 서로
지켜줘야 할 규칙을 정함

API 명세서

프론트엔드와 백엔드 간에 주고받는

API에 대해 명시

Wiki 작성






- 프로젝트 일정 정리
- 매일 진행되는 스크럼 내용 기록
(전날 회고 & 당일 목표)
- 오피스아워 진행 내용 기록
- 파트 별 README 문서 작성

PROJECT 진행과정

2021
12

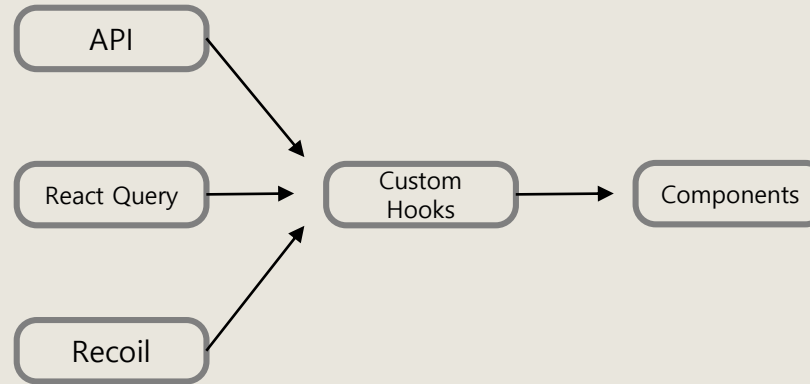
2022
01

SUN	MON	TUE	WED	THU	FRI	SAT
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15

-  Sprint 1
-  Sprint 2
-  Sprint 3
-  Sprint 4
-  Presentation

PROJECT 파트 별 주요 기술 : 프론트엔드

Custom Hooks



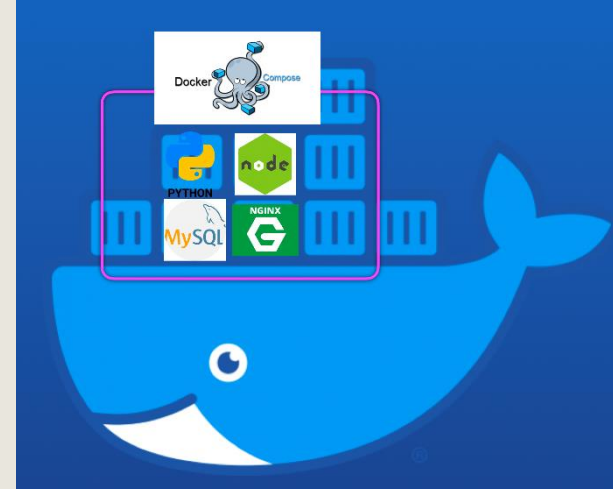
```
✓ hooks
JS useContent.js
JS useContentListQuery.js
JS useInput.js
JS useResult.js
JS useUserPick.js
```

State Management



PROJECT 파트 별 주요 기술 : 백엔드

Docker



API

| User API

POST	api/user/signup 사용자 회원가입
POST	api/user/signin 사용자 로그인
GET	api/user/signout 사용자 로그아웃
GET	api/user/isSignin 사용자 로그인 되어있는지 확인
POST	api/user/delete 사용자 삭제

| Contents API

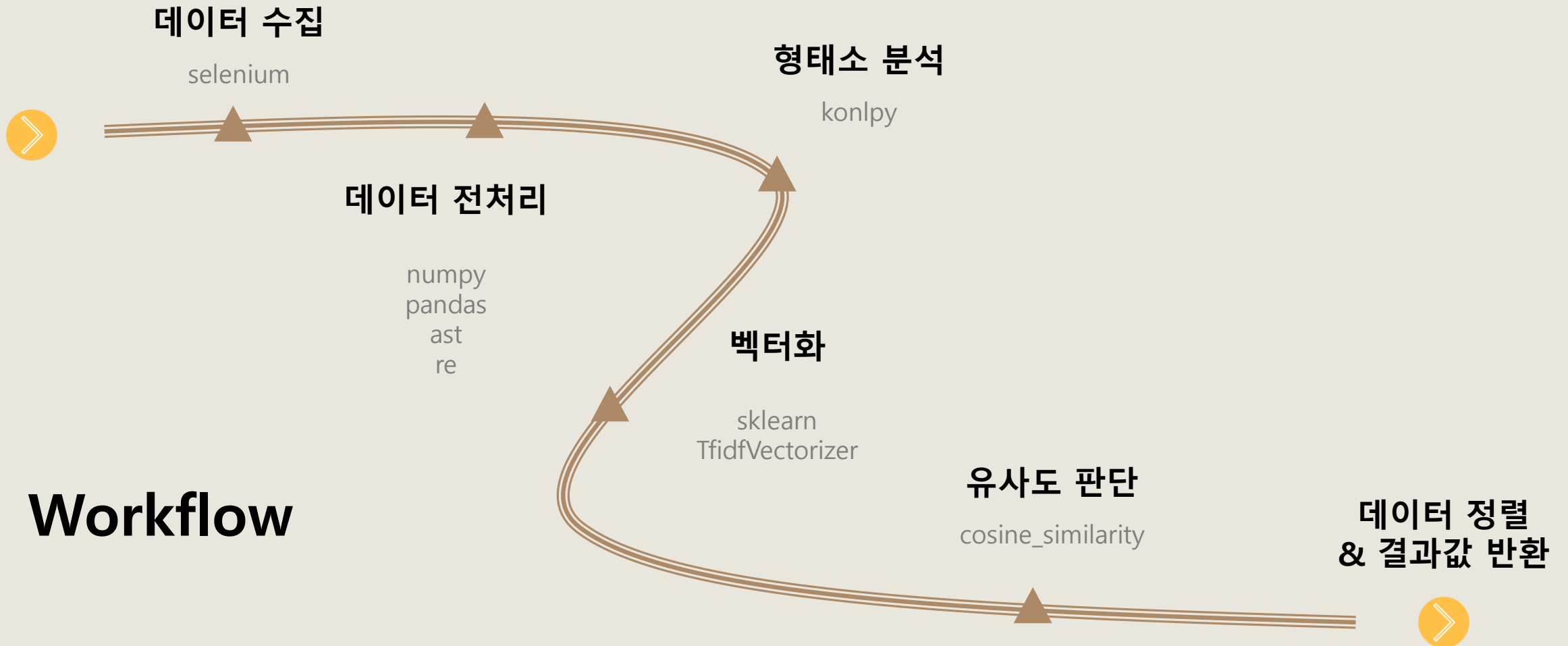
POST	api/contents/list 유저가 선택한 genre와 연도 기반으로 선택되는 영화 리스트
POST	api/contents/recommend 사용자에게 영화 추천 및 OTT 추천
GET	api/contents/detail/{id} 영화의 세부 페이지 정보
GET	api/contents/search 키워드와 타입을 기반으로 검색되는 영화 리스트
GET	api/contents/userpick 유저가 찮한 영화들의 목록
POST	api/contents/userpick 유저가 찮한 콘텐츠를 DB에 반영
GET	api/contents/favorite 유저들이 좋아하는 영화 10개

추천 알고리즘

• Content Based Filtering

영화의 특징을 벡터화 하여 거리가 가까운 영화를 추천해주는 알고리즘

- 특징 : 감독, 장르, 출연진, 시놉시스
- Tf-idf Vectorizer를 사용하여 특징을 벡터화
- Cosine Similarity를 사용하여 가장 가까운 벡터 반환
- 가중치평점과 추가적인 알고리즘을 사용하여 다양한 영화 추천



TF-IDF

Term Frequency Inverse Document Frequency
단어 빈도 - 역 문서 빈도

$$TF - IDF = TF(t, d) \times IDF(t)$$

- TF(t, d) : Term Frequency
특정 문서 d에서 특정 단어 t 등장 횟수
- DF(t) : Document Frequency
특정 단어 t가 등장한 문서의 수
- IDF(t) : Inverse Document Frequency
문서 빈도의 역수

Cosine Similarity

특징 A, B의 벡터값이 각각 주어졌을 때 코사인 유사도

$$\text{Similarity} = \cos(\theta)$$

$$= \frac{a \cdot b}{\|a\| \|b\|}$$

$$= \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

가중치 평점

단순한 평점이 아닌 전체 평균점수와,
평가한 사람의 수가 반영된 점수를 사용하여
영화를 추천

$$\text{가중치 평점} = \frac{v}{v+m} R + \frac{m}{v+m} C$$

- v: 개별 영화에 평점을 투표한 횟수
- m: 평점을 부여하기 위한 최소 투표 횟수
- R: 개별 영화에 대한 평점
- C: 전체 영화에 대한 평균 평점

선택된 영화가 다음과 같다면?



1. 가장 비슷한 영화 추천

Cosine similarity와 가중치 평점이 가장 높은 영화



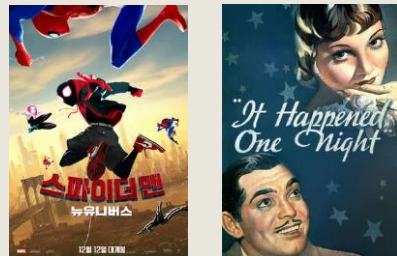
2. 공통된 영화 추천

중복적으로 추천되는 영화들은 가중치를 부여하여 우선적으로 추천



3. 시놉시스가 유사한 영화 추천

감독, 출연진, 장르 등이 달라도 시놉시스의 내용이 비슷한 영화를 추천

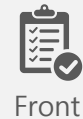


FIN 문제점 & 해결방안

문제점

추천결과 페이지에서 화면 전환을 하고 다시 돌아갔을 때, 윈도우에 다시 포커스 했을 때 추천 결과가 달라지는 문제

화면 전환시
추천 결과가 바뀜



전역 상태로 저장

해결방안

추천 알고리즘이 인풋은 같더라도 결과에는 랜덤한 내용도 들어가기 때문에 결과 페이지에 진입했을 때 데이터 통신을 하게 되면 결과가 계속 바뀌게 된다. 데이터 통신을 페이지 진입 이전에 하고 추천 결과를 전역 상태로 저장을 하여 결과페이지에 보여주게 함으로써 문제를 해결하였다

user api 관련 Test code 작성 시, user session 유지의 어려움이 있었다

Test code client의
필요성



Flask test_client

flask의 test_client의 기능과 python의 with문을 활용하여, 현재 flask app의 test client를 생성하고 그 client가 with문 안에서 session을 유지할 수 있도록 하였다

약 2만개의 영화의 cosine_similarity 매트릭스(20000*20000)의 모델의 크기가 약2.7G 라는 규모로 만들어져 서버에서 동작하지 않은 문제가 발생하였다

Cosine similarity
매트릭스의 크기



매트릭스 분할

추천 알고리즘에 2만등까지 유사한 영화 정보가 사용되지 않음으로, 알고리즘에 필요한 60등까지의 순위만 파일로 저장하여 이를 DB에서 가져와 바로 추천하는 방법으로 효율적으로 모델을 구성하였다



FE

김현하

이번 프로젝트 이전까지는 혼자서 프론트엔드 부분만 개발하는 경험만 있었습니다. 프로젝트를 통해서 백엔드와 같이 협업을 해보면서 **원활한 프로젝트 진행을 위해서는** 프론트에 관련된 지식뿐만 아니라 서버가 어떻게 구조되어있는지, 회원가입/로그인 기능이 어떻게 진행되는지 등의 **백엔드 지식 또한 필요하다는 것을 느꼈고** 이번 프로젝트를 통해 백엔드 팀원들의 자세한 설명을 통해 조금씩 알게 되었던 시간이었습니다.

저에게 이번 프로젝트는 **다양하고 새로운 도전**을 해볼 수 있는 경험이었습니다. 제 노력은 눈으로 보이지 않지만, 안정적으로 서비스가 동작되는 것을 보고 보람을 느꼈습니다.



BE

장서우

처음으로 팀프로젝트를 하면서 이전 개인 프로젝트와는 또 다른, 백엔드에 조금 더 집중할 수 있는 시간이 되었던 것 같습니다. 이 과정에 익숙해지는 데에 시간이 걸려 어려움이 있었지만, 이번 프로젝트를 계기로 정말 많은 것을 배울 수 있었습니다. 특히, Front와 Back이 어떻게 api를 가지고 서로 소통하는지와 Docker, API 문서, 테스트 코드, CI/CD 등을 공부하면서 **Backend의 처음부터 끝까지 어떤 과정으로 진행되는지** 감을 잡을 수 있었던 시간이었습니다.

처음으로 다른 포지션의 팀원과 프로젝트를 진행하며 **다른 포지션과의 의사소통 방법을 많이 배웠습니다.** 같은 데이터 분석 포지션이면 설명에 생략하는 부분이 있거나 상대 설명을 쉽게 이해하고 넘어가지만, 다른 포지션 설명은 보다 집중하며, 부족하면 다시 설명을 부탁하고, 데이터 분석 설명을 보다 쉽게 이해할 수 있도록 의사소통에 노력한 것 같습니다.



DB

전현호

데이터 분석 파트의 일원으로 팀 프로젝트가 시작되면서 내가 엘리스 교육과정에서 **배운 내용을 얼마나 이해하고 있을까**에 대한 궁금증을 가지고 프로젝트에 임했습니다. 데이터를 수집과 정제, 시각화하는 방법에 익숙해지고 **추천 알고리즘에 대해 새롭게 공부하는 기회**가 되었습니다. 또한 팀 프로젝트는 개인의 역량에 더해 **팀원과의 의사소통과 다른 파트에 대한 이해도**가 정말 중요함을 깨달았습니다. 함께해준 팀원분들과 코치님들 고생 많으셨고 감사드립니다!



이무원

BE



이재근
(팀장)

DB

