

FeeLM

FeeL? FiLM? FeeLM!

서울 4반 10조 박수연 이예원

CONTENTS

FeeeeeeeeeeeLM

01

개요

02

기술 스택

03

기능 설명

04

시연

05

느낀점 및 후기

06

Q&A



Feel + *Film*



icon

FeelLM

main logo

개발 기간

2024.11.14 ~ 2024.11.26

FeeLM 역할 분담

이 예 원

박 수 연

FRONT
END

FRONT↔BACK 연결
컴포넌트 개발
API 통신 (TMDB 등)
토큰 인증 권한 관리
UI / UX 설계
CSS 디자인

BACK
END

DB 및 DUMMY DATA 생성
모델 설계
기능 구현
-회원
-영화
추천 알고리즘

FeeLM 기술 스택

FRONT



BACK



FeeLM 개발 일정

	11.14 (THU) ~ 11.17 (SUN)				11.18 (MON) ~ 11.24 (SUN)							11.25 (MON) ~ 11.27 (WED)		
	11.14	11.15	11.16	11.17	11.18	11.19	11.20	11.21	11.22	11.23	11.24	11.25	11.26	11.27
기능 명세서 ERD	FRONT & BACK													
LOGO 및 디자인				FRONT										
모델 설계				FRONT & BACK										
회원 및 영화기능					BACK				BACK					
추천 기능										BACK				
영화 데이터 관리	FRONT													
API 통신		FRONT												
UI / UX					FRONT									
컴포넌트 기능 구현								FRONT						

FeeLM 기능

회원 기능

회원가입
로그인 / 로그아웃
회원정보 변경

영화 기능

영화
-전체 / 상세
-북마크 / 좋아요

리뷰 기능

리뷰
-리뷰 CRUD
-리뷰 좋아요
-평점

추천 기능

영화 기반 추천

제목, 장르, 키워드 등
가중치를 조절하며
특정 영화와의
유사도 측정

북마크 기반 추천

사용자가 북마크한
영화들을 하나로 통합
다른 영화들과
유사도 측정

가중치 순위
제목, 장르, 키워드

리뷰 평점 기반 추천

사용자가 리뷰를 작성한
영화들의 평점 고려

높은 평점을 줬다
= 즐거리가 좋다!

가중치 순위
제목, 즐거리

```

# 북마크 기반 추천
def calculate_combined_weighted_similarity_bookmark(movies_df, bookmark, field1, field2):
    bookmark = pd.DataFrame(bookmark)

    # 북마크된 영화들의 특성을 결합
    combined_bookmark = bookmark.apply(
        lambda row: "{} {}".format(' '.join([str(row[field1]) if pd.notna(row[field1]) else '' for field1 in field1]),
                                     ' '.join([str(row[field2]) if pd.notna(row[field2]) else '' for field2 in field2]))
    )

    # 북마크된 모든 영화의 특성을 하나로 결합
    combined_bookmark = pd.Series([' '.join(combined_bookmark.values)])

    # 전체 영화 데이터의 특성을 결합
    combined_features = movies_df.apply(
        lambda row: "{} {}".format(' '.join([str(row[field1]) if pd.notna(row[field1]) else '' for field1 in field1]),
                                     ' '.join([str(row[field2]) if pd.notna(row[field2]) else '' for field2 in field2]))
    )

    # 북마크 특성을 전체 특성에 추가
    combined_features = pd.concat([combined_features, combined_bookmark], ignore_index=True)

    # 벡터화 및 유사도 계산
    count_vect = CountVectorizer(min_df=1, ngram_range=(1, 2))
    combined_mat = count_vect.fit_transform(combined_features)
    combined_sim = cosine_similarity(combined_mat, combined_mat)

    return combined_sim

def find_sim_movie_combined_bookmark(df, sorted_ind, bookmark, top_n=10):
    # 입력된 영화의 인덱스 찾기
    movie = pd.DataFrame(bookmark)
    title_movie = pd.DataFrame()
    title_indexes = []
    for title in movie['title']:
        matched_movie = df[df['title']==title]
        title_movie = pd.concat([title_movie, matched_movie], ignore_index=True)
        title_indexes.extend(df[df['title']==title].index.tolist())

    # 리뷰 평점 기반 추천
    user_reviews = user.reviews.values('movie_id', 'rating')
    movie_id = []
    tmdb_id = []
    rating = []
    for review in user_reviews:
        movie_id.append(review['movie_id']-1)
        rating.append(review['rating'])
        tmdb = Movie.objects.get(id=review['movie_id'])
        tmdb_id.append(tmdb)

    rating_df = pd.DataFrame({'movie_id':movie_id, 'tmdb_id':tmdb_id, 'rating':rating})

    if rating_df.empty:
        rating_rec = movies_df[movies_df['vote_avg'] >= 7].sample(n=20)[['tmdb_id', 'title']]
        rating_recom = Movie.objects.filter(tmdb_id__in=rating_rec['tmdb_id']).tolist()
    else:
        rating_rec = movie_recommendation_system_combined_rating(
            "C:/Users/SSAFY/Desktop/SF12_Feelm/pjt_movie/django-pjt/movies/fixtures/movietop1.json",
            rating_df,
            'title', 'overview', 'production_com', 'original_lang', 'genre', 'keyword',
            4, 4, 1, 1, 3, 2.5,
            20
        )
        rating_recom = Movie.objects.filter(tmdb_id__in=rating_rec)

    # 북마크 기반 추천
    bookmark_list = list(request.user.bookmark.all().values())
    # DataFrame 생성
    bookmark = pd.DataFrame(bookmark_list)

    if bookmark.empty:
        movies_rec = movies_df[movies_df['vote_avg'] >= 7].sample(n=20)[['tmdb_id', 'title']]
        movies_recom = Movie.objects.filter(tmdb_id__in=movies_rec['tmdb_id']).tolist()
    else:
        movies_rec = movie_recommendation_system_combined_bookmark(
            "C:/Users/SSAFY/Desktop/SF12_Feelm/pjt_movie/django-pjt/movies/fixtures/movietop1.json",
            bookmark,
            'title', 'overview', 'production_com', 'original_lang', 'genre', 'keyword',
            5, 2, 1, 3, 4, 4,
            20
        )
        movies_recom = Movie.objects.filter(tmdb_id__in=movies_rec)

    serializer_bookmark = MovieListSerializer(movies_recom, many=True)
    serializer_rating = MovieListSerializer(rating_recom, many=True)
    return Response({'review_recommendations':serializer_rating.data, 'bookmark_reccomendations': serializer_bookmark.data})

```




시연 

FeeLM 느낀점 / 후기

이 예 원

FE와 BE를 함께 하는 프로그램을 경험할 수 있어서 좋았다.
이번 프로젝트를 바탕으로 다음 프로젝트를 진행할 때는 컴포넌트 파일의 구성을 미리 그려놓고 진행해야겠다고 느꼈다.

박 수 연

관통 프로젝트를 통해 백엔드 프레임워크를 다루며 어색했던 부분을 익숙하게 만들었다.
BE 개발을 맡았기에 FE 개발 담당과 끊임없는 소통이 굉장히 중요하다는 것을 깨달았던 기회가 되었다.



Q&A

자유롭게 질문해주세요!



감사합니다!