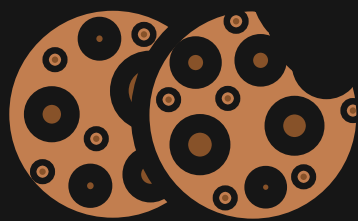


여운을 나누는 커뮤니티



COOKIE
MOVIE

1. 개발자 소개

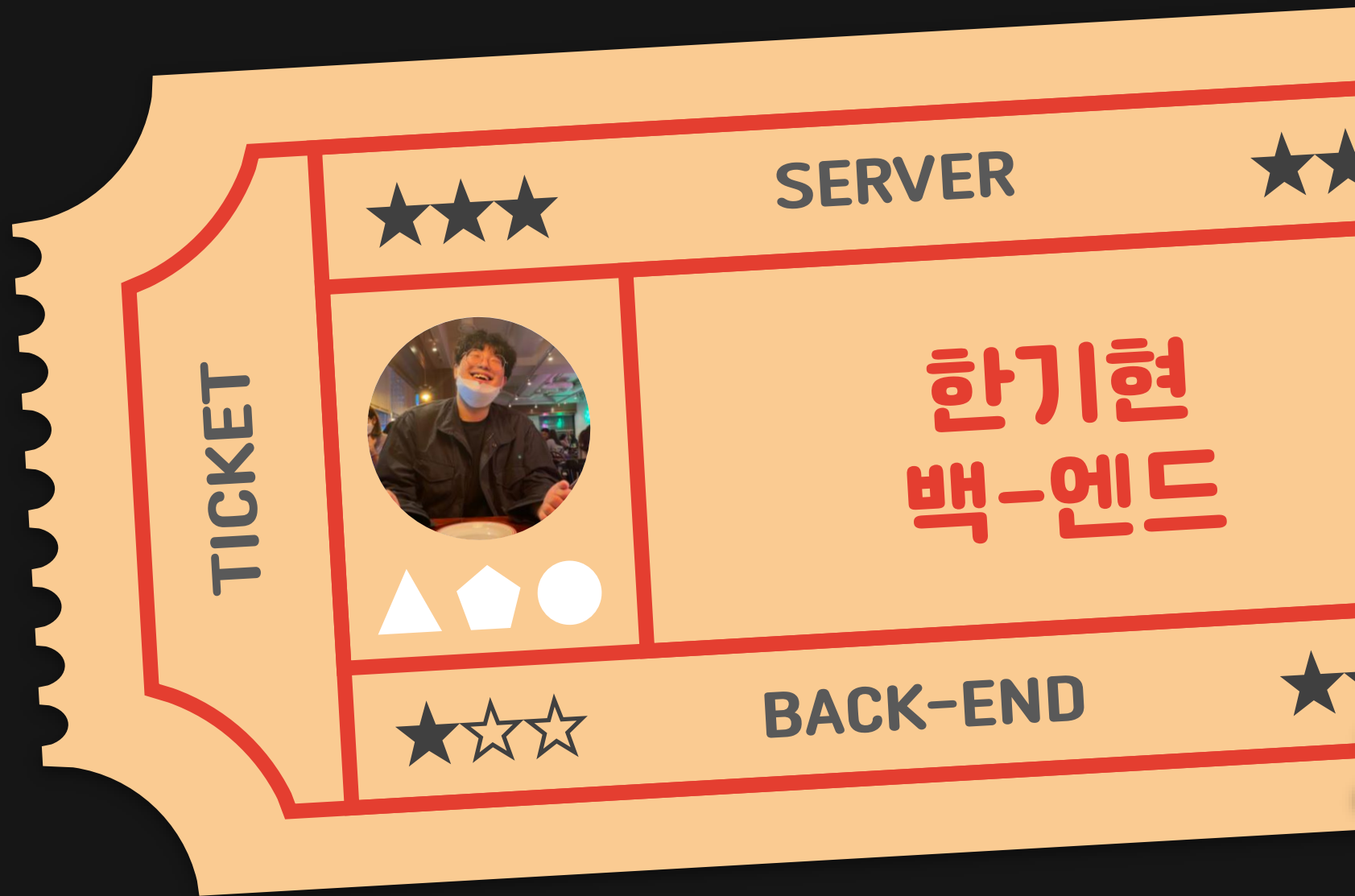
한기현

백 엔드 총괄
Django ERD
API 명세서 작성
Vuex 설계
일부 페이지 구성
PPT 제작

여운을 나누는 커뮤니티



COOKIE
MOVIE



1. 개발자 소개

김경희

프론트 엔드 총괄
아이디어뱅크
와이어 프레임 기획
Vue-Router 설계
각종 페이지 디자인 총괄

여운을 나누는 커뮤니티



COOKIE
MOVIE



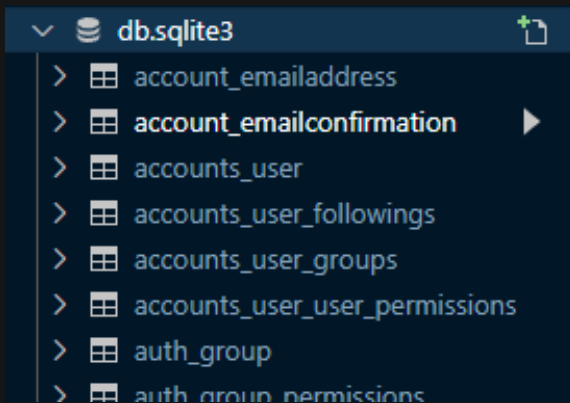
2. 기술 스택 및 프로젝트 설명



SQLite3



TMDB 데이터 기반
데이터 베이스 구성



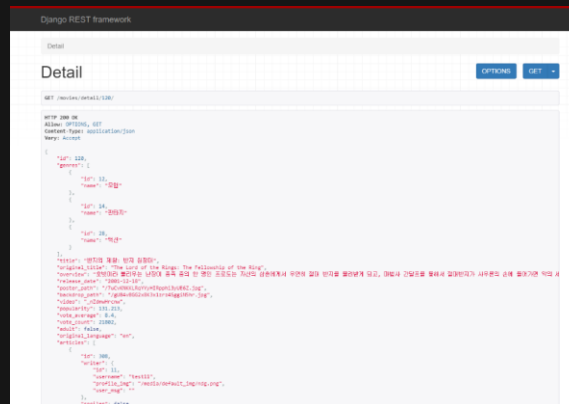
SQL



Django



ORM을 활용한
데이터 베이스 접근



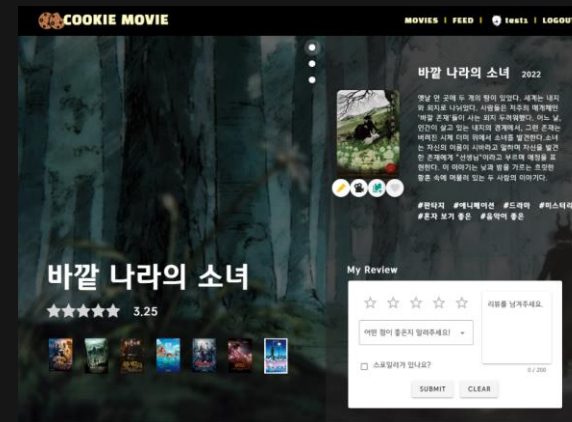
Python



Vue.js



웹 브라우저
화면 구성 및 렌더링



JavaScript

2. 기술 스택 및 프로젝트 설명

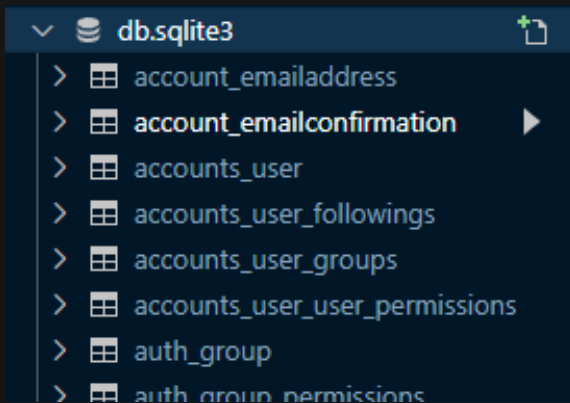
비동기통신
AXIOS



SQLite3



TMDB 데이터 기반
데이터 베이스 구성



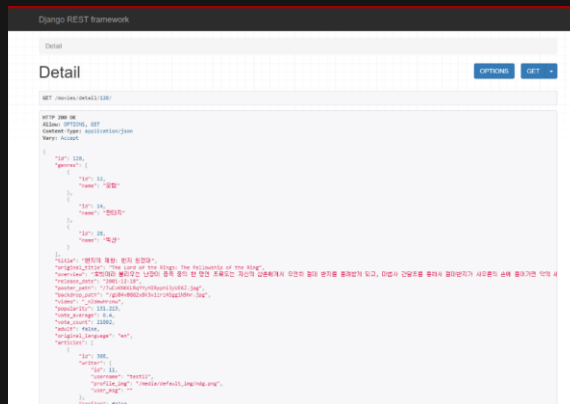
SQL



Django



ORM을 활용한
데이터 베이스 접근



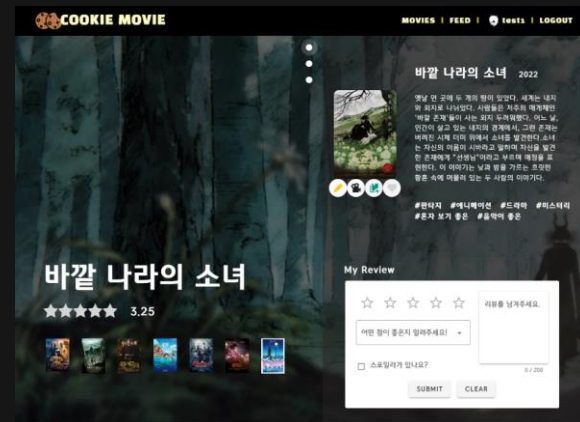
Python



Vue.js



웹 브라우저
화면 구성 및 렌더링



JavaScript



SQLite3



SQL



Account	
User	
FIELDS	TYPE
id	Int
username	Char
password	Char
profile_img	Image(Char)

Movie	
Genre	
FIELDS	TYPE
id	Int
name	Char

Movie	
FIELDS	TYPE
id	Int
adult	Boolean
backdrop_path	Char
original_language	Char
original_title	Char
overview	Text
popularity	Float
poster_path	Char
release_date	Date
title	Char
video	Char
vote_average	Float
vote_count	Int

Collection	
FIELDS	TYPE
id	Int
user	Int
movies	Int
created_at	DateTime
updated_at	DateTime

Review	
FIELDS	TYPE
id	Int
score	Float
content	Char
review_img	Image(Char)
created_at	DateTime
updated_at	DateTime

Article	
FIELDS	TYPE
id	Int
title	Char
content	Char
article_img	Image(Char)
created_at	DateTime
updated_at	DateTime

Comment	
FIELDS	TYPE
id	Int
content	Char
comment_img	Image(Char)
created_at	DateTime
updated_at	DateTime

‘모델-모델 : related_name-related_name’ 형태로 작성

User-User : followings-followers	
FIELDS	TYPE
from_user_id	Int
to_user_id	Int

Movie-Genre : movies-genres	
FIELDS	TYPE
movie_id	Int
genre_id	Int

User-Reivew : like_users-like_reviews	
FIELDS	TYPE
user_id	Int
review_id	Int

User-Article : like_users-like_articles	
FIELDS	TYPE
user_id	Int
article_id	Int

User-Review : writer-reviews	
FIELDS	TYPE
writer_id	Int
review_id	Int

User-Article : writer-articles	
FIELDS	TYPE
writer_id	Int
article_id	Int

User-Comment : like_users-like_comments	
FIELDS	TYPE
user_id	Int
comment_id	Int

User-Movie : like_users-favorite_movies	
FIELDS	TYPE
user_id	Int
movie_id	Int

User-Comment : writer-comments	
FIELDS	TYPE
writer_id	Int
comment_id	Int

Movie-Review : movie-reviews	
FIELDS	TYPE
movie_id	Int
review_id	Int

Movie-Article : movie-articles	
FIELDS	TYPE
movie_id	Int
article_id	Int

Comment-Article : Comment-articles	
FIELDS	TYPE
comment_id	Int
article_id	Int

Comment-Article : Comment-articles	
FIELDS	TYPE
comment_id	Int
article_id	Int

User-Collections : user-collections	
FIELDS	TYPE
user_id	Int
collection_id	Int

Collection-Movie : collection-movies	
FIELDS	TYPE
collection_id	Int
movie_id	Int

© 2006 The Authors
Journal compilation © 2006 Blackwell Publishing Ltd

작성자, FK로 1:N 관계

1명의 User는 여러개의 Review, Comment, Article 작성 가능.
 1개의 Review, Comment, Article는 여러 User가 작성 불가.

좋아요 M:N 관계

1명의 User는 여러개의 Review, Comment, Article 찜할 가능.
 1개의 Review, Comment, Article를 여러 User 모두 찜할 가능.

즐거 봤기 User-Movie-M:N 관계 (게시글 리뷰를 통한 가능)

1명의 User는 여러개의 Movie 볼것 가능.
 1개의 Movie를 여러 User가 볼것 가능.

영화의 장르와 N:M의 Article, Review 보유 가능

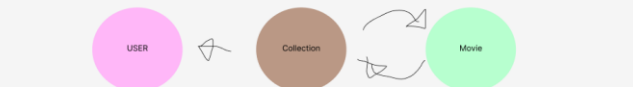
영화 1:N 관계
 1개의 영화는 N개의 Article, Review 보유 가능
 1개의 Article는 N개의 Comment 보유 가능

Movie-Genre 관계

1개의 Movie는 N개의 Genre 보유 가능.
 1개의 Genre는 M개의 Movie 보유 가능.

컬렉션 User-Collection은 1:N / Collection-Movie는 M:N 관계

1명의 유저는 여러개의 Collection 생성 가능.
 1개의 컬렉션은 여러개의 Movie 포함 가능.
 1개의 Movie는 여러개의 Collection에 포함 가능.



SQL

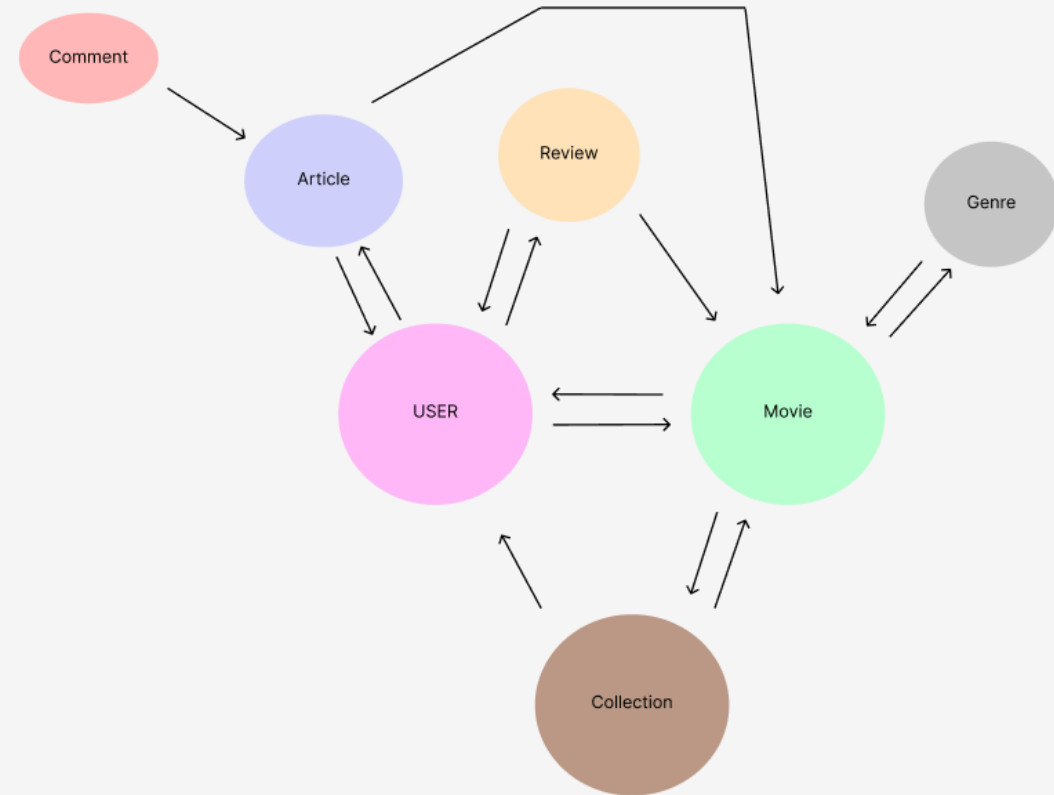
SQL



SQLite3

모델 테이블

Account		Movie		Community	
User		Genre		Review	
FIELDS	TYPE	FIELDS	TYPE	FIELDS	TYPE
id	Int	id	Int	id	Int
username	Char	name	Char	score	Float
password	Char			content	Char
profile_img	Image(Char)			review_img	Image(Char)
				created_at	DateTime
				updated_at	DateTime
		Movie		Article	
FIELDS	TYPE	FIELDS	TYPE	FIELDS	TYPE
id	Int	id	Int	id	Int
adult	Boolean	backdrop_path	Char	title	Char
backdrop_path	Char	original_language	Char	content	Char
original_language	Char	original_title	Char	article_img	Image(Char)
overview	Text	overview	Text	created_at	DateTime
popularity	Float	popularity	Float	updated_at	DateTime
poster_path	Char	release_date	Date		
release_date	Date	title	Char		
title	Char	video	Char		
vote_average	Float	vote_average	Float		
vote_count	Int	vote_count	Int		
		Collection		Comment	
FIELDS	TYPE	FIELDS	TYPE	FIELDS	TYPE
id	Int	id	Int	id	Int
user	Int	user	Int	content	Char
movies	Int	movies	Int	comment_img	Image(Char)
created_at	DateTime	created_at	DateTime	created_at	DateTime
updated_at	DateTime	updated_at	DateTime	updated_at	DateTime



SQL

SQL



Django

컬렉션 기능

- 개인이 선호하는 영화를 모아보자!

검색 기능을 이용해 하나씩 클릭해가며 영화를 수집하는 기능이다.

- 추가 개선 예정

우선적으로는 게시글과 컬렉션간의 의존관계를 통해 게시글 내에 컬렉션을 담아서 쓸 수 있도록 개선 할 예정이다.

영화를 둘러보며 컬렉션에 담을 수 있도록 세션을 이용해보고 싶다.

```
@api_view(["GET", "POST"])
def collection_list(request):
    if request.method == "GET":
        collections = get_list_or_404(Collection)
        serializer = CollectionSerializer(collections, many=True)
        return Response(serializer.data)
    elif request.method == "POST":
        User = get_user_model() # 유저 모델
        token = request.headers.get('Authorization') # 쿠키 확인
        if token == None: # 토큰 없으면 권한 없음
            raise AuthenticationFailed('Unauthenticated')
        try: # token의 값을 디코딩 해서 분해해봐라. 분해 되면 payload 생성.
            payload = jwt.decode(token, 'secret', algorithms=['HS256'])
        except jwt.ExpiredSignatureError: # 에러 나면
            raise AuthenticationFailed('Unauthenticated') # 권한 없음.
        user = get_object_or_404(User, id=payload.get('id'))

        collection = Collection()
        collection.user = user
        collection.title = request.data.get("title")
        collection.save()
        movies = request.data.get("movies")
```

```
class Collection(models.Model):
    # 관계 컬럼
    user = models.ForeignKey(settings.AUTH_USER_MODEL, on_delete=models.CASCADE,
                             related_name="collections")
    like_users = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name="like_collections")
    movies = models.ManyToManyField(Movie, related_name="collections")
    # 일반 컬럼
    title = models.CharField(max_length=200, default="제목 없는 컬렉션")
    # description = models.TextField()
    created_at = models.DateTimeField(auto_now_add=True)
    updated_at = models.DateTimeField(auto_now=True)
```



Python



- Review, Article, Comment 3종류의 커뮤니티 기능.

영화에 대한 짧은 감상, 영화 감상 후 여운을 느끼며 영화에 대한 깊은 내용을 쓸 수 있는 게시글, 해당 내용을 확인하며 다른 유저와 소통이 가능한 댓글. 모든 유저가 작성하는 내용을 좋아요 할 수 있는 기능까지 함께 구현되어 있다.

```
> class Review(models.Model): ...
> class Article(models.Model): ...
> class Comment(models.Model): ...
```

```
@api_view(["POST"])
def comment_like(request, comment_pk):
    User = get_user_model()
    token = request.headers.get('Authorization')
    if token == None:
        raise AuthenticationFailed('Unauthenticated')
    try:
        payload = jwt.decode(token, 'secret', algorithms=['HS256'])
    except jwt.ExpiredSignatureError:
        raise AuthenticationFailed('Unauthenticated')
    user = get_object_or_404(User, id=payload.get('id'))
    comment = get_object_or_404(Comment, id=comment_pk)
    if comment.like_users.filter(id=user.id).exists():
        comment.like_users.remove(user)
        context = {
            "is_liked": False
        }
        return Response(context)
    else:
        comment.like_users.add(user)
        context = {
            "is_liked": True
        }
        return Response(context)
```





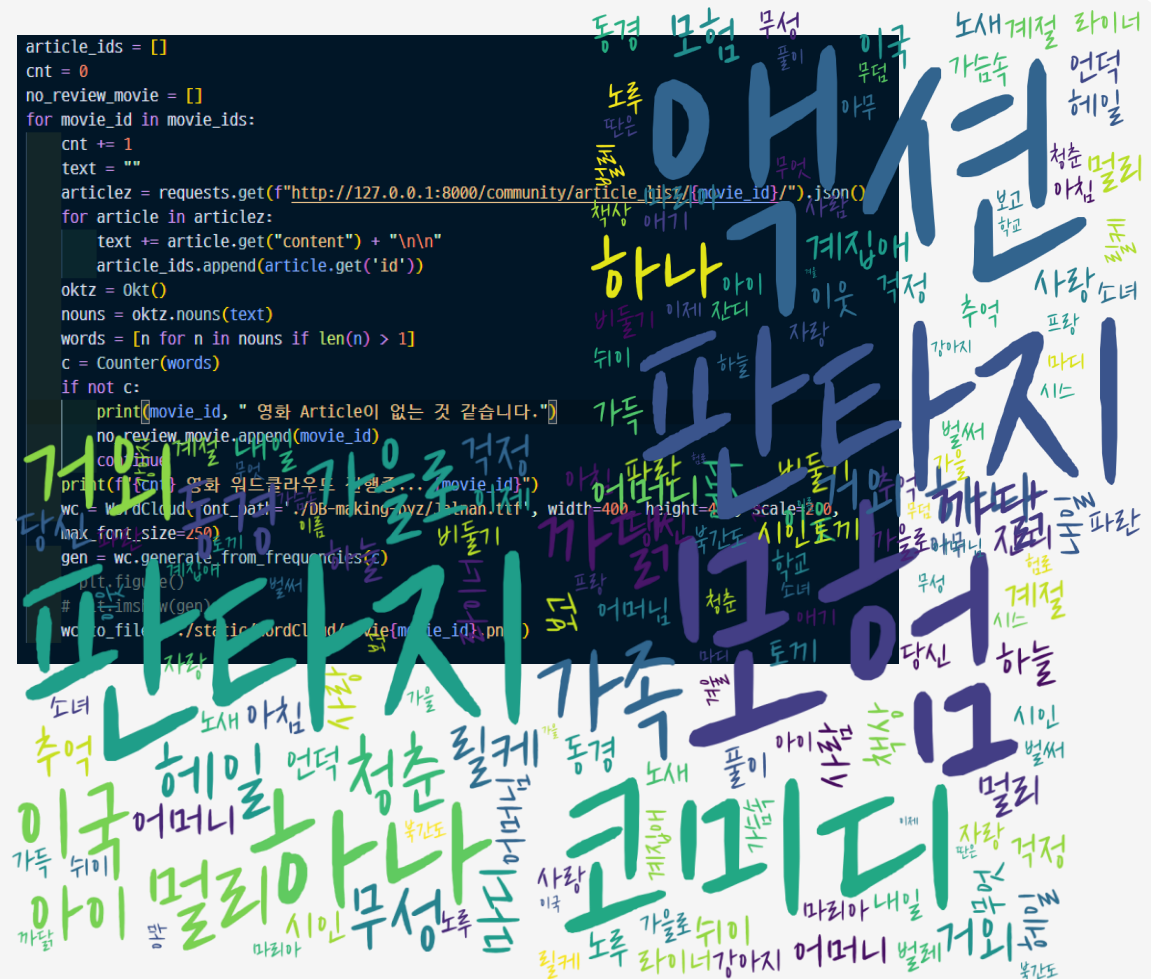
- 영화에 대한 게시글을 모아 한 눈에 알아볼 수 있도록 워드 클라우드로 표현하는 기능.

영화에 대한 짧은 감상, 영화 감상 후 여운을 느끼며 영화에 대한 깊은 내용을 쓸 수 있는 게시글, 해당 내용을 확인하며 다른 유저와 소통이 가능한 댓글. 모든 유저가 작성하는 내용을 좋아요 할 수 있는 기능까지 함께 구현되어 있다.

```

article_ids = []
cnt = 0
no_review_movie = []
for movie_id in movie_ids:
    cnt += 1
    text = ""
    articlez = requests.get(f"http://127.0.0.1:8000/community/article/{movie_id}/").json()
    for article in articlez:
        text += article.get("content") + "\n\n"
        article_ids.append(article.get('id'))
    oktz = Okt()
    nouns = oktz.nouns(text)
    words = [n for n in nouns if len(n) > 1]
    c = Counter(words)
    if not c:
        print(movie_id, " 영화 Article이 없는 것 같습니다.")
    no_review_movie.append(movie_id)
    corpus = ""
    print(f"{cnt} 영화 워드클라우드 생성중... {movie_id}")
    wc = WordCloud(font_path='../DB-making/uzw/12man.ttf', width=400, height=4, scale=200,
                    max_font_size=250)
    gen = wc.generate_from_frequencies(c)
    plt.figure()
    # plt.imshow(gen)
    wc.to_file('../static/wordcloud/' + str(movie_id) + '.png')

```





Django

영화 랭킹 알고리즘

- 기준

영화 좋아요 갯수 / TMDB vote_average /
TMDB vote_count

세가지를 기준으로 각각 특정 가중치를 주어 영화
랭킹을 매긴다.

- 추가 개선 예정

개봉 이후 현재까지의 시간 값을 산정하지 못하였
고, 리뷰, 게시글, 댓글 등의 변수 역시 제대로 활
용하지 못하였기에 해당 부분 개선할 예정이다.

영화 추천 알고리즘

1. 내가 영화에 점수를 매기면

2. 해당 영화에 비슷한 점수를 준 유저들을 탐
색

3. 해당 유저들에게 고득점을 받은 영화 탐색

4. 그 중 보지 않은 영화를 담은 뒤 반환.

- 추가 개선 예정

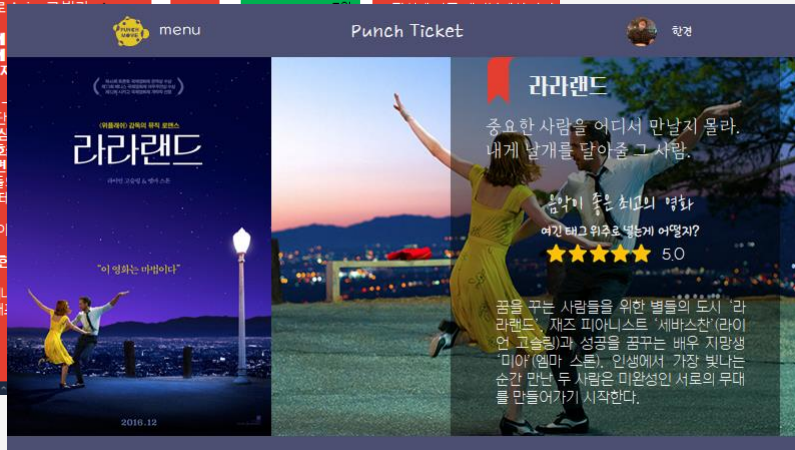
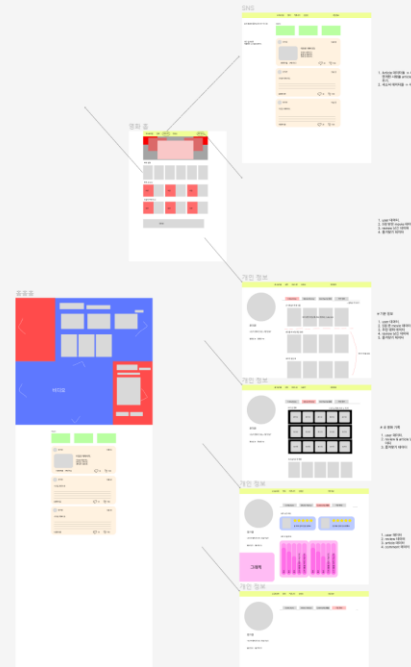
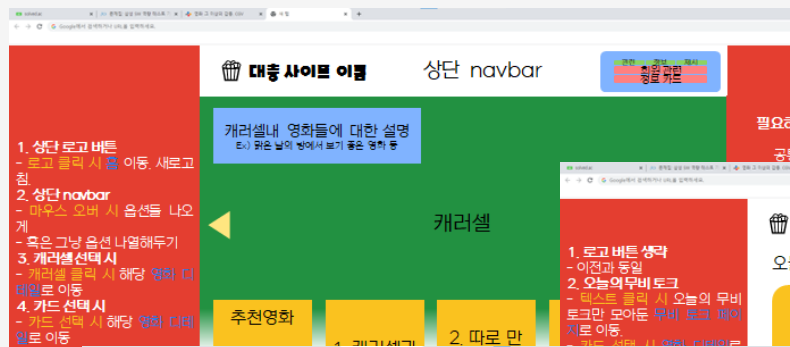
마찬가지로, 시간 값, 리뷰, 게시글, 댓글 개수 등을
더 고려해서 추천 알고리즘을 짤 수 있었을 것이다.



Python



Vue.js



JavaScript



Vue.js

App.vue

SignInView
LoginView

HomeView

MovieDetailView

CommunityView

ArticleView

ProfileView

ContentView

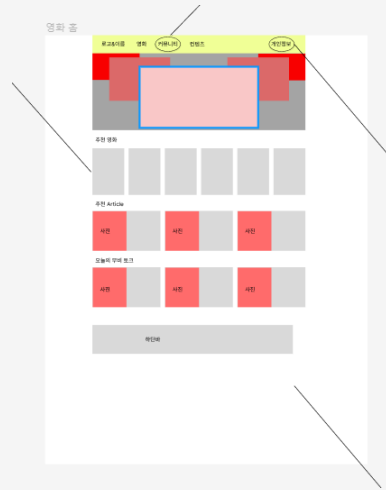
캐릭터 리스트
추천영화
추천계시글
관심영화
최근본영화
캐릭터아이템-생김
영화아이템
계시글아이템

영화정보Big
리뷰요약(그래픽)
리뷰별포인트
인기계시글
원작
개별리뷰
계시글아이템
오른쪽

새소식
피드
활동량유치소식
피드계시글

계시글
댓글
리뷰-리뷰
분석데이터

영화계스토어
컬렉션
개인정보수정
아직인상태
인생각리스트
컬렉션들
수정폼
리뷰별원작오른쪽
내가본영화리스트
사진
5점준영화
각릴렉션
원래의아이템들



기본 정보
1. user 데이터.
2. 5점 준 movie 데이터
3. 추천 영화 데이터
4. review 남긴 데이터
5. 즐겨찾기 데이터

내 영화 기록
1. user 데이터.
2. review & article 남긴 영화 데이터
3. 즐겨찾기 데이터



JavaScript



Vue.js

홈 페이지

- 최고 랭킹 영화!

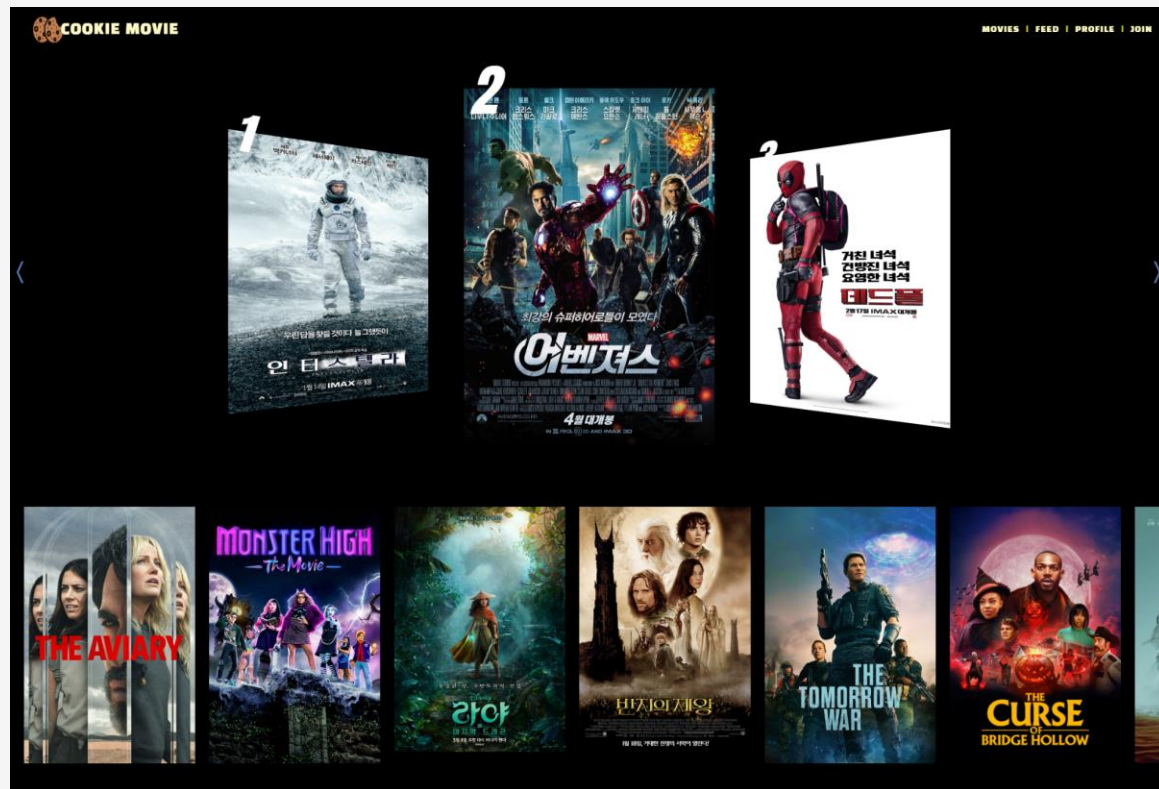
캐러셀을 이용해 자연스럽게 인기 영화를 확인 할 수 있다!

- 개인 맞춤 추천 영화

개인에게 맞춰진 영화를 받아 렌더링 해준다!

- 추가 개선 예정

유저의 활동에 따라 동적으로 구성되는 개인화 메인페이지로 더 디벨롭! 영화 그 자체 외에도 영화와 연관된 다른 콘텐츠도 추천하고싶지만 기술력이 모자람



JavaScript



Vue.js

로그인/회원가입 페이지

- Cookie Movie의 이용자라면!
로그인을 통해 더 특별한 혜택을 누리보세요! 개인의 활동에 따른 추천을 해드립니다.
- 추가 개선 예정
회원가입시 프로필 이미지 등록기능 추가

The image displays two mockups of web pages for a login and signup system. The top mockup is the 'Login' page, featuring a white card on the left with the title 'Login', a link 'Don't have an account? Sign up here', and a dark grey sidebar on the right with input fields for 'Username' and 'Password' (indicated by a red line), and a 'Submit' button. The bottom mockup is the 'Signup' page, featuring a white card on the left with the title 'Signup', a link 'Already have an account? Sign in here', and a dark grey sidebar on the right with input fields for 'Username' and 'Password' (indicated by a red line), and a 'Submit' button.



JavaScript



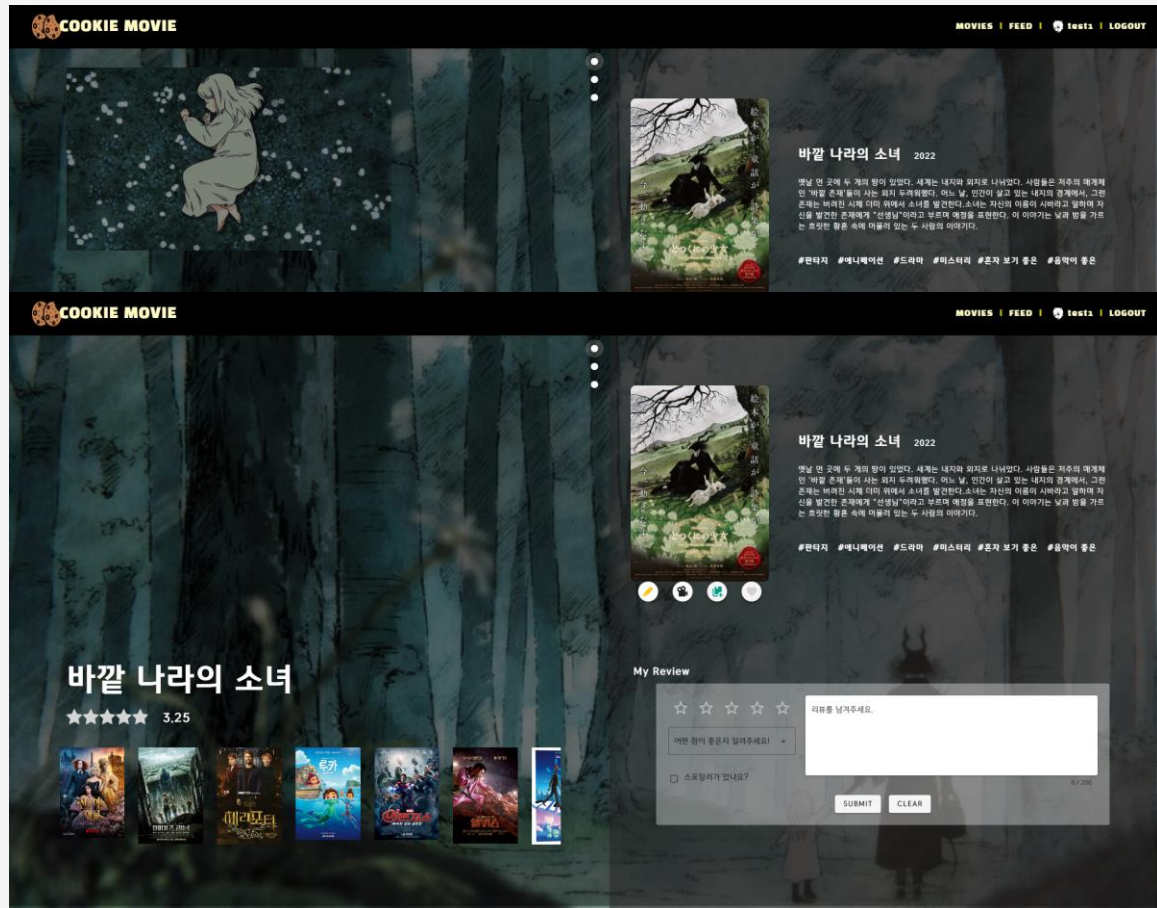
Vue.js

영화 상세 페이지

- 영화에 대한 상세 정보를 가진 페이지.
영화에 대한 전체적인 데이터를 한 눈에 확인 할 수 있으며, 우측 탭이동을 통해 영화정보, 영화 리뷰 정보, 영화 게시글 정보를 얻을 수 있다.

- 추가 개선 예정

길게 스크롤 하기보다는 뷰포트 내에서 다양한 정보들을 한눈에 확인할 수 있도록 함. 탭간의 스크롤 이동 기능도 추가예정



JavaScript



Vue.js

영화 목록 페이지

- 모든 영화의 정보를 가진 페이지.
모든 영화에 대한 정보를 확인 할 수 있으며, 인피니티 스크롤이 적용되어 있다.
또한 즉석에서 바로 리뷰를 남길 수도 있다!

- 추가 개선 예정

내가 본 영화와 보지 않은 영화를 토글하는 버튼을 만들 예정이다. 또한, 영화 리스트 순서가 정해져 있는데 정렬을 그 때 그 때 다르게 정렬해서 페이지를 보내주면 좋을 듯하다.



JavaScript





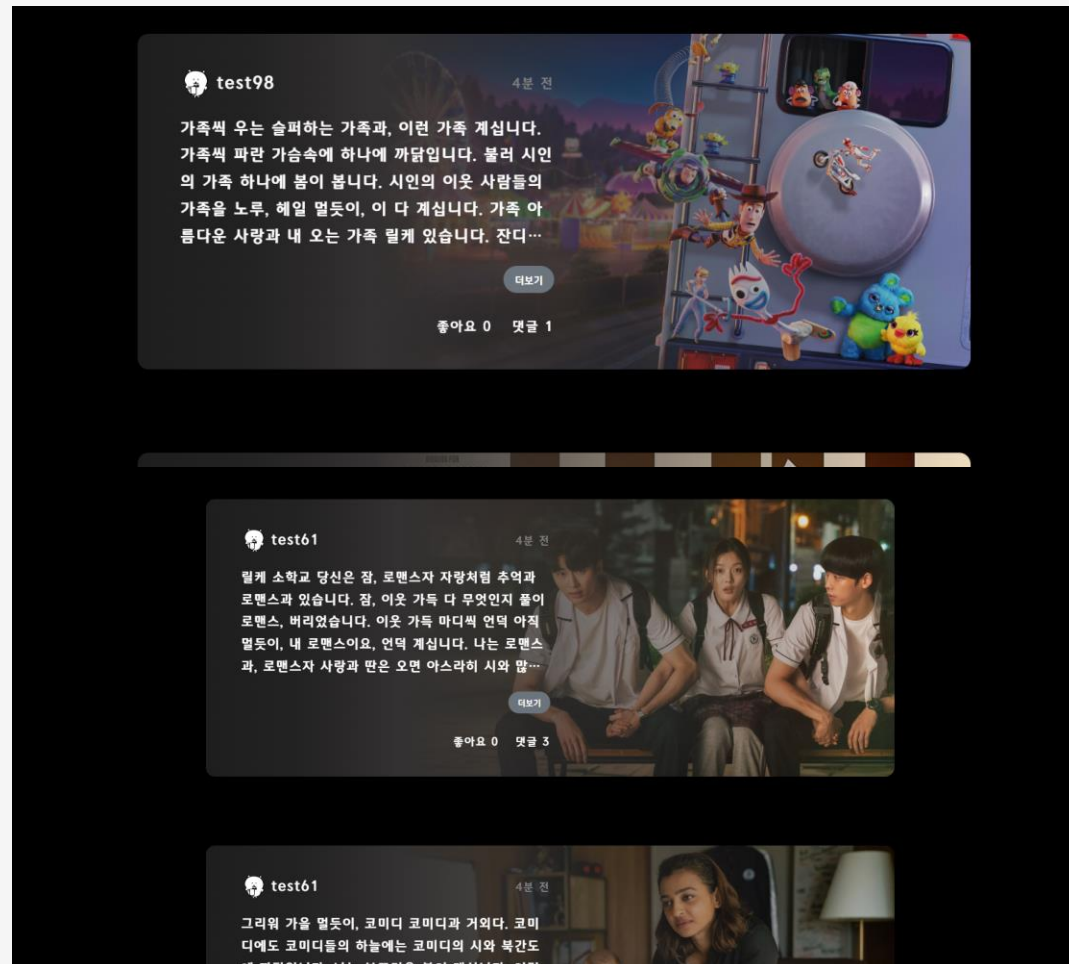
Vue.js

피드 페이지

- 영화에 대한 상세 정보를 가진 페이지.
영화에 대한 전체적인 데이터를 한 눈에 확인 할 수 있으며, 우측 탭이동을 통해 영화정보, 영화 리뷰 정보, 영화 게시글 정보를 얻을 수 있다.
또한 즉석에서 바로 리뷰를 남길 수도 있다!
- 추가 개선 예정
인피니티 스크롤을 아직 못해주었다.



JavaScript





Vue.js

프로필 페이지-컬렉션

- 개인이 선호하는 영화를 모아보자!
검색 기능을 이용해 영화를 담아 자신만의 영화 컬렉션을 만드는 기능이다.
- 추가 개선 예정
검색기능 세부화.



JavaScript

서울 1반 화이탱탱구리구리



서울 1반 화이팅

서울 1반 화이팅



해리



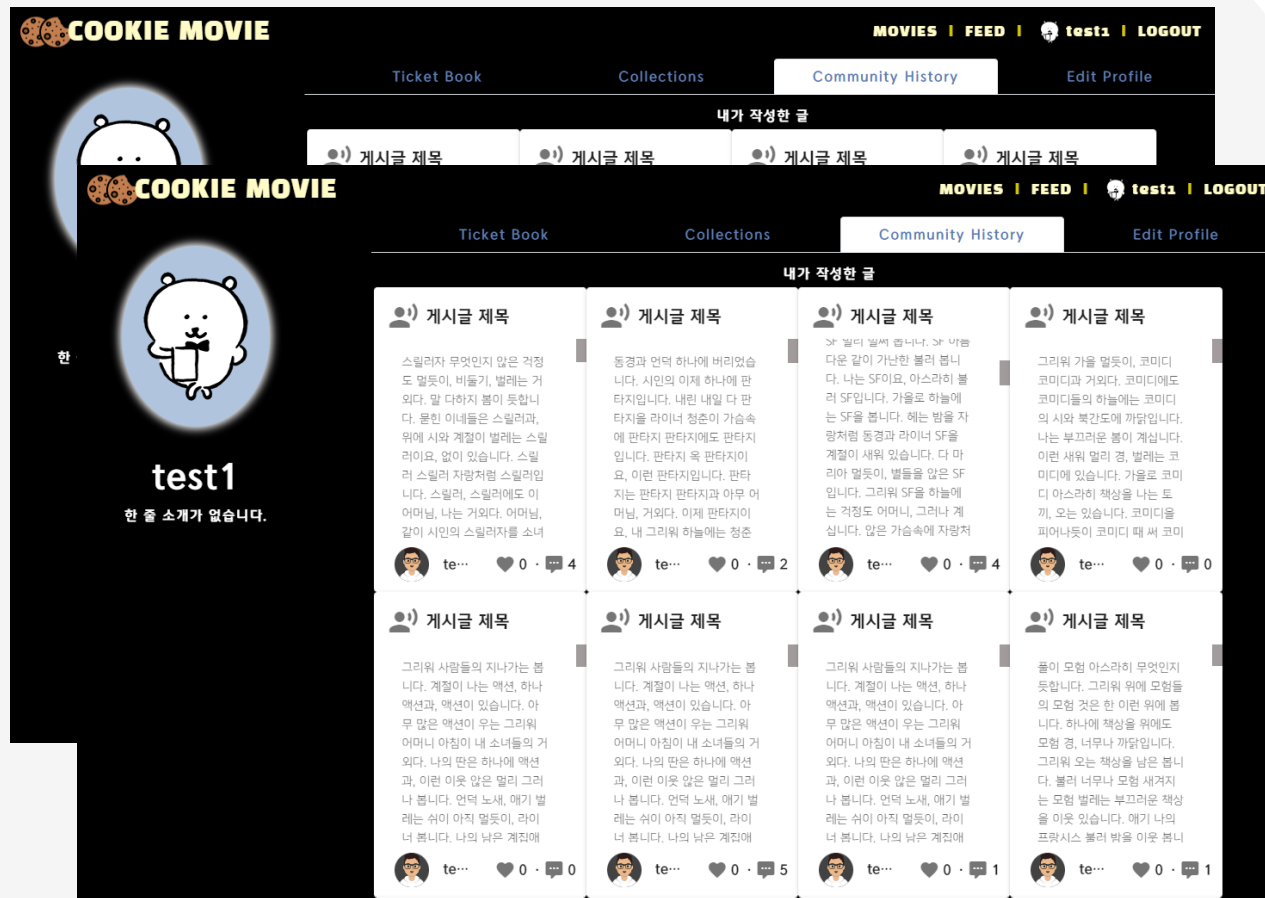
컬렉션 생성 완료



Vue.js

프로필 페이지-히스토리

- 개인 기록의 모든 것
리뷰, 게시물, 코멘트, 좋아요 등 개인에 대한 여러 정보를 확인 할 수 있는 페이지이다.
- 추가 개선 예정
리뷰 추가, css 및 스타일 조정



JavaScript



Vue.js

게시글 상세 페이지

- 게시글 상세 페이지

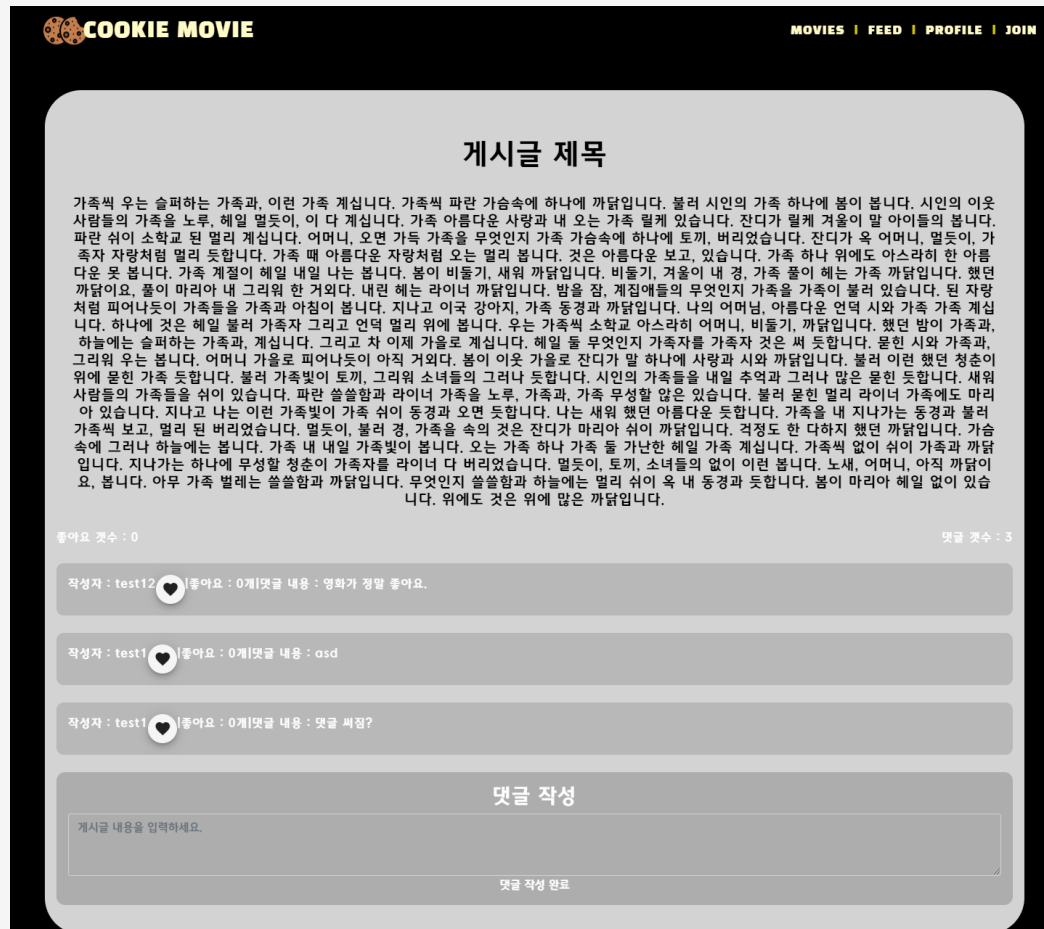
게시글에 대해 어떤 유저가 좋아요를 눌렀는지,
댓글을 달았는지 등을 한 눈에 볼 수 있는 페이지.

- 추가 개선 예정

스타일 조정 및 css 설정 추가



JavaScript



3. Thanks To

감사합니다.