

데이터베이스 Term Project 최종 결과 보고서

분 반	3 분반		
작 품 명	Hrror movie review blog		
개발기간	20 년 5 월 1 일 ~ 20 년 6 월 21 일		
지도교수	컴퓨터공학과	오병우	
구 분	학년	학 번	성 명
제출자	3	20200370	김혜진
<p>본인은 데이터베이스 Term Project 최종 결과 보고서를 첨부와 같이 제출합니다. 제출한 보고서는 본인이 직접 개발 및 작성하였으며, 거짓이나 부정이 있다면 F학점을 받고 학칙에 의거하여 처벌받겠습니다 (학적부에 등재).</p> <p style="text-align: right;">2022 년 6 월 21 일</p> <p style="text-align: right;">제 출 자 김혜진</p> <p style="text-align: center;">컴퓨터공학심화프로그램</p>			

최종 결과 보고서

Horror movie review blog

0. 결과 요약

(1) 3개 테이블 Join 수행한 결과를 HTML 테이블 형태로 출력 구현 완료

- △. 3개 테이블이 아니라 2개 테이블(movie, review)만 조인하였다.

(2) 게시판 또는 블로그 형태 템플릿 사용

- ○. 게시판이나 블로그 형태 템플릿을 직접 쓴 것이 아니라 부트스트랩의 컴포넌트와 레이아웃을 조합하는 형식으로 구현하였다. 따라서 아래는 주로 사용한 컴포넌트와 레이아웃을 소개한다.
- Grid 레이아웃 (<https://getbootstrap.com/docs/5.2/layout/grid/>)



A diagram showing three adjacent rectangular boxes, each labeled 'Column', representing a three-column grid layout.

Column	Column	Column
--------	--------	--------

[그림 1] 부트스트랩 도큐먼트에서 소개하는 그리드 레이아웃 모습

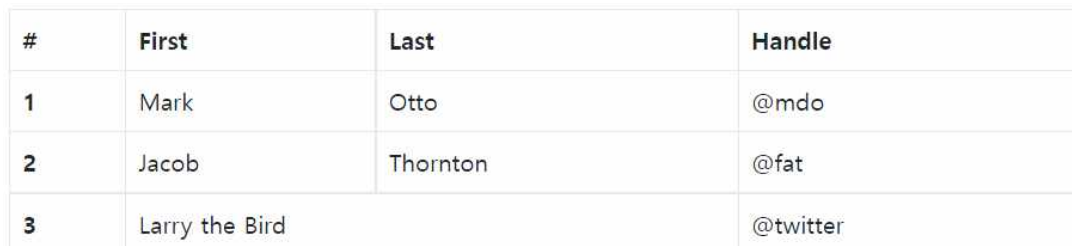


A diagram showing a horizontal menu bar with four rounded rectangular buttons. The buttons contain the text: '리뷰 목록 보기', '리뷰 안 된 영화 목록 보기', '전체 영화 목록 보기', and '영화 등록하기'.

리뷰 목록 보기	리뷰 안 된 영화 목록 보기	전체 영화 목록 보기	영화 등록하 기
-------------	-----------------------	----------------	-------------

[그림 2] 메뉴 바를 그리드 레이아웃으로 구현한 모습

- Table 콘텐츠 (<https://getbootstrap.com/docs/5.2/content/tables/>)



A table with 4 columns: #, First, Last, and Handle. It contains 3 rows of data.

#	First	Last	Handle
1	Mark	Otto	@mdo
2	Jacob	Thornton	@fat
3	Larry the Bird		@twitter

[그림 3] 부트스트랩 도큐먼트에서 소개하는 테이블 콘텐츠 모습

제목(개봉 년도)	감독	포스터
인시디어스 (2010)	제임스 완	
한줄평	굿굿	
평점	4.4	

[그림 4] 리뷰 내용을 테이블로 구현한 모습

- Button 컴포넌트 (<https://getbootstrap.com/docs/5.2/components/buttons/>)



[그림 5] 부트스트랩 도큐먼트에서 소개하는 버튼 컴포넌트 모습



[그림 6] 리뷰 등록/삭제 바로가기를 버튼으로 구현한 모습

- Form (<https://getbootstrap.com/docs/5.2/forms/form-control/>)

Email address

Example textarea

[그림 7] 부트스트랩 도큐먼트에서 소개하는 폼 모습

닉네임

비밀번호

댓글 내용

[그림 8] 댓글 작성을 폼으로 구현한 모습

- 이 외에도 텍스트 정렬, 마진이나 패딩 속성 등을 부트스트랩 클래스로 구현하였다.

(3) 게시물 및 댓글 삽입/삭제/변경/검색 기능 구현

- 게시물 삽입○ 삭제○ 변경○ 검색○

• 게시물 삽입

영화 등록

영화 제목	<input type="text" value="그것"/>
개봉년도	<input type="text" value="2017"/>
감독	<input type="text" value="안드레스 무시에티"/>
포스터	<input type="text" value="https://movie-phinf.pstatic.net/20170904_146/1504487646916G9OQb_JP"/>

[그림 9] 영화 등록 페이지. 등록 버튼을 누르면 데이터베이스의 movie 테이블에 삽입된다.

리뷰 등록



장화, 홍련(2003) - 김지운

한줄 평	<input type="text" value="스토리가 조금 이상해요 이해를 못 하겠어요"/>
평점	<input type="text" value="2.0"/>
명장면	<input type="text" value="https://movie-phinf.pstatic.net/20111222_76/13245361633317n05Q_JPEC"/>


[그림 10] 리뷰 등록 페이지. 리뷰 안 된 영화 목록이나 전체 영화 목록에서 등록 버튼을 누르면 해당 영화에 대한 리뷰를 작성하여 review 테이블에 삽입된다.

• 게시물 삭제

리뷰 안 된 영화 목록

순번	제목(개봉년도)	감독	포스터	등록/삭제
1	인시디어스(2010)	제임스 완		<input type="button" value="등록"/> <input type="button" value="삭제"/>

[그림 11] 리뷰 안 된 영화 목록 테이블에서 삭제버튼을 눌러 삭제할 수 있다.

명장면	
<div style="text-align: right;"> 수정 삭제 </div>	

[그림 12] 리뷰의 한줄평을 클릭했을 때 나오는 리뷰 디테일뷰에서 삭제 버튼을 눌러 리뷰를 삭제할 수 있다.

• 게시글 변경

리뷰 수정

한줄 평	예시
평점	4.5
명장면	https://movie-phinf.pstatic.net/20111222_76/13245361633317n05Q_JPEC
목록 수정	

[그림 13] 그림 12의 수정 버튼을 눌러 리뷰를 수정할 수 있다.

• 게시글 검색

전체 영화 목록

순번	제목(개봉년도)	감독	포스터	등록
1	인시디어스(2010)	제임스 완		등록
2	장화, 홍련(2003)	김지운		등록

제목 검색

검색

[그림 18] 전체 영화 목록과 제목으로 검색할 수 있는 입력란의 모습

- 댓글 삽입○ 삭제○ 변경○ 검색○

• 댓글 삽입

댓글 목록		
작성자	내용	수정/삭제
안녕	최고!	등록
•		

[그림 19] 리뷰 디테일 뷰 하단의 댓글 목록에서 댓글을 작성하는 모습

• 댓글 삭제

댓글 목록		
작성자	내용	수정/삭제
안녕	11	수정 삭제

[그림 20] 댓글 목록에서 삭제 버튼을 눌러 댓글을 삭제할 수 있다.

• 댓글 수정

localhost:3000 내용:
수정할 내용을 입력하세요

22

확인

취소

localhost:3000 내용:
비밀번호를 입력하세요

확인

취소

[그림 21] 댓글 수정 버튼을 누르면 뜨는 입력 창. 수정 내용과 비밀번호를 입력하여 댓글을 수정할 수 있다.

• 댓글 검색

댓글 목록		
작성자	내용	수정/삭제
안녕	최고!	수정 삭제
안녕	최고2	수정 삭제

닉네임	댓글 내용	등록
비밀번호		
작성자로 댓글 검색		검색

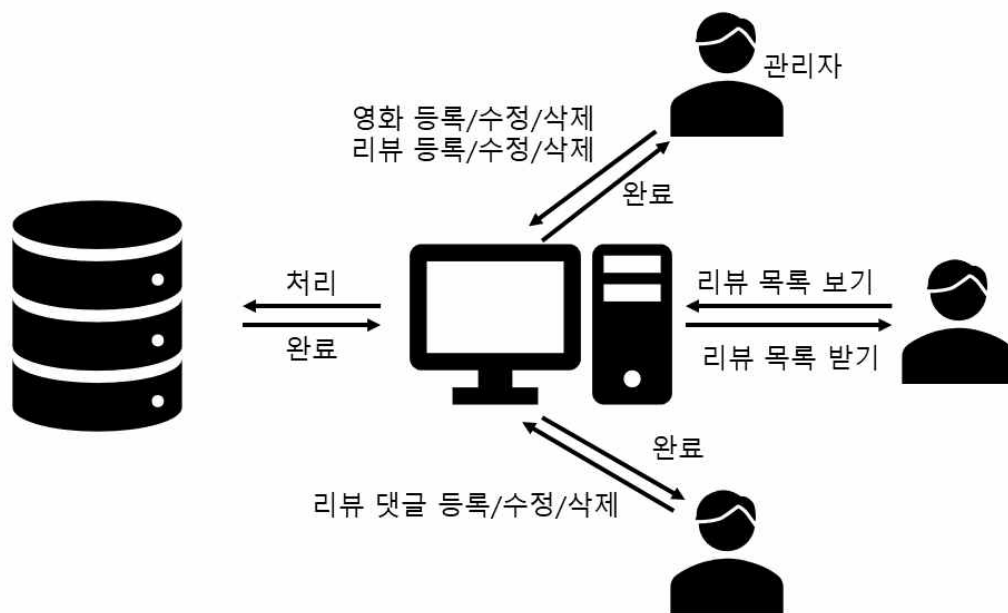
[그림 22] 댓글 등록 밑의 검색란에 작성자를 입력하여 댓글을 검색할 수 있다.

1. 서론

공포영화를 좋아하는데, 내가 본 공포영화들이 어떤 것이 있었는지 잘 기억이 나지 않을 때가 많아 안타까웠다. 사진을 이용하면 다시 그 장면을 보았을 때 더욱 기억에 잘 남고, 다른 사람들에게 내가 감상한 평가 내용과 평점을 공유하고 싶었다. 다른 사람에게 보여주고 공포영화 선택에 도움을 주려는 목적도 있지만 개인적으로도 기록용으로 남기고 싶어 공포영화 리뷰 블로그를 구상하게 되었다.

2. 작품 개요

2.1 전체 구성도



[그림 23] 공포 영화 리뷰 블로그 구조

그림 28은 공포 영화 리뷰 블로그의 전체 그림이다. 관리자는 관리자의 비밀번호를 이용하여 영화와 리뷰를 등록/수정/삭제할 수 있다. 사용자는 리뷰 목록이나 영화 목록을 요청할 수 있으며 영화의 경우 검색도 가능하다. 각 리뷰에 대한 댓글도 등록/수정/삭제할 수 있으며 작성자 명으로 검색도 가능하다. 서버는 유저의 요청을 서비스하기 위해 데이터베이스에 처리 요청을 내리며 그 결과를 반환한다.

2.2 설계구성요소 및 설계제한 요소

<설계구성요소 및 설계제한 요소>

설계 구성요소						설계 제한요소						
목표 설정	합성	분석	구현/제작	시험/평가	결과 도출	성능	규격/표준	경제성	미학	신뢰성	안정성/내구성	환경
○		○	○				○		○	○		○

- 목표 설정: 웹을 기반으로 자신이 좋아하는 취미에 대한 게시판을 작성하고 로그인 기능, 게시물에 대한 댓글 기능을 제공하는 홈페이지를 개발한다. 영화 등록/삭제, 리뷰 등록/수정/삭제는 비밀번호를 아는 관리자만이 등록할 수 있게 하여 관리자의 정보 기록장 역할을 한다.
- 분석: 취미 DB를 분석한다.
- 구현/제작: 데이터베이스를 정규화 규칙 BCNF에 기반하여 제작한다.

(1) 규격/표준

- 규격/표준: HTML, JSON 사용, 테이블 3개 이상 사용, Join 연산 사용
- 사용하는 테이블
 - movie: 영화 코드, 제목, 개봉년도, 감독, 포스터 이미지, 영화 등록 날짜를 저장한다.
 - review: 리뷰 코드, 영화 코드, 한줄 평, 명장면 사진, 평점, 리뷰 등록 날짜를 저장한다.
 - comment: 댓글 코드, 리뷰 코드, 닉네임, 비밀번호, 댓글 내용, 댓글 등록 날짜를 저장한다.
- Join 연산
 - movie, review: 영화 코드로 join 하여 해당 영화에 등록된 리뷰들을 출력한다.

(2) 미학

- 미학: 미려하고 직관적인 사용자 인터페이스 설계, Image 출력 (웹 상의 이미지를 주소 복사하여 데이터베이스에 등록하고, 이미지 태그의 src 속성에 바로 이용한다.)
- 템플릿: Bootstrap을 사용한다. Bootstrap의 그리드 레이아웃, 테이블, 버튼, 입력 폼을 주로 사용한다.

(3) 신뢰성

- 신뢰성: DB를 반영하여 웹페이지로부터 데이터 삽입(영화 등록, 리뷰 등록, 댓글 등록) 기능, 데이터 삭제(영화 삭제, 리뷰 삭제, 댓글 삭제) 기능, 데이터 변경(리뷰 수정, 댓글 수정) 기능, 데이터 검색(영화 검색, 댓글 검색) 기능을 구현했다.
- 환경: 웹 환경에서 React, Express, MySQL, JSON 사용하였다.

(4) 환경

- 환경: 웹 환경에서 React, Express 사용하였다.

3. 설계

3.1 Use Case

3.1.1 관리자 영화 등록

- 관리자는 영화의 제목, 개봉년도, 감독 이름, 포스터 이미지 주소를 작성한다.
- (설계서에는 영화를 네이버 API로 검색할 수 있게 하여 실세계에 존재하는 영화만 등록하는 기능을 설계했는데, 시간 상 구현하지 못했다.)
- 대신 관리자만 작성할 수 있다는 점을 들어 관리자가 실세계에 존재하는 영화만을 작성할 것이라고 가정했다.
- 등록 버튼을 누르면 알림 창으로 비밀번호를 입력한다. 비밀번호가 일치하는 경우에만 등록된다.

3.1.2 관리자 영화 삭제

- 3.1.7의 [리뷰 안 된 영화 리스트 보기] 창에서 삭제 버튼을 눌러 영화를 삭제할 수 있다. 3.1.8의 [전체 영화 리스트 보기]에는 리뷰가 달린 영화도 나오기 때문에 영화 삭제 시 리뷰가 달리지 않은 영화만 삭제하기 위해서 이렇게 구현했다.
- 삭제 버튼을 누르면 알림 창으로 비밀번호를 입력한다. 비밀번호가 일치하는 경우에만 삭제된다.
- (삭제 시 정말로 삭제하시겠습니까? 라는 문구를 넣는 것으로 설계했는데, 모든 삭제 기능에서 비밀번호를 입력하세요. 라는 문구로 대체하였다.)

3.1.3 관리자 리뷰 등록

- 3.1.7의 [리뷰 안 된 영화 리스트 보기]나 3.1.8의 [전체 영화 리스트 보기]에서 등록 버튼을 눌러 해당 영화에 대한 리뷰를 등록할 수 있다.
- 등록 버튼을 누르면 선택한 영화의 정보(제목, 감독, 개봉년도, 포스터)가 위에 뜨고, 밑에는 관리자가 직접 한 줄 평, 명장면 사진, 평점을 입력한다.
- 등록 버튼을 누르면 알림 창으로 비밀번호를 입력한다. 비밀번호가 일치하는 경우에만 등록된다.

3.1.4 관리자 리뷰 수정

- 3.1.10의 [리뷰 보기]에서 수정 버튼을 눌러 해당 리뷰를 수정할 수 있다.
- 수정 버튼을 누르면 등록과 같은 화면이 나오되 기존 내용이 작성된 채로 나오기 때문에 관리자는 필요한 내용만 수정하여 등록할 수 있다.
- 수정 버튼을 누르면 알림 창으로 비밀번호를 입력한다. 비밀번호가 일치하는 경우에만 등록된다.

3.1.5 관리자 리뷰 삭제

- 3.1.10의 [리뷰 보기]에서 삭제 버튼을 눌러 해당 리뷰를 삭제할 수 있다.
- 삭제 버튼을 누르면 알림 창으로 비밀번호를 입력한다. 비밀번호가 일치하는 경우에만 삭제된다.

3.1.6 리뷰 리스트 보기

- (설계서에는 메인화면에서 리뷰 리스트 보기와 3.1.7의 [리뷰 안 된 영화 리스트 보기]를 일부분 띄워놓고 해당 페이지로 이동하는 기능을 구현하려고 설계했지만, 시간 상 구현하지 못했다.)

-
- 리뷰가 리뷰의 평점 순서대로 테이블로 출력된다.
 - 테이블 내용에는 영화 정보(제목, 개봉년도, 감독, 포스터)와 리뷰 정보(한줄 평, 평점, 명장면)이 나온다.
 - 테이블에서 한줄 평이 있는 셀을 클릭하면 해당 리뷰에 대한 디테일 뷰(3.1.10의 [리뷰 보기] 화면)이 나온다.

3.1.7 리뷰 안 된 영화 리스트 보기

- (설계서에는 페이지를 나누어 일정 개수만 출력하였고 리스트의 정렬 순서를 변경할 수 있는 기능을 설계했지만, 시간 상 구현하지 못했다.)
- 영화는 등록됐으나 리뷰가 되지 않은 영화의 리스트를 볼 수 있다.
- 해당 영화의 정보(제목, 개봉년도, 감독, 포스터)가 테이블로 출력된다.
- 테이블 각 행의 마지막 열에는 등록/삭제 버튼을 만들어 해당 영화에 대한 리뷰를 등록하거나 영화를 DB에서 삭제할 수 있다.

3.1.8 전체 영화 리스트 보기

- (설계서에는 없지만 한 영화에 여러 리뷰가 달릴 수 있다는 점을 고려하여 추가된 내용이다.)
- 리뷰의 유무에 상관없이 DB에 등록된 전체 영화 리스트를 볼 수 있다.
- 해당 영화의 정보(제목, 개봉년도, 감독, 포스터)가 테이블로 출력된다.
- 테이블 각 행의 마지막 열에는 등록 버튼을 만들어 해당 영화에 대한 리뷰를 등록할 수 있다.
- 영화 제목으로 영화를 검색할 수 있다.

3.1.9 영화 검색

- (설계서에는 3.1.7의 [리뷰 안 된 영화 리스트 보기]에서 영화 제목으로 검색할 수 있도록 하였지만, 영화 목록에서 영화를 검색하는 기능이 맞는 것 같아 수정한 내용이다.)
- 3.1.8의 [전체 영화 리스트 보기]와 같은 화면의 하단에서 영화 제목으로 영화를 검색할 수 있다. 그러면 해당 리스트에서 검색 결과 리스트로 테이블 내용이 변경된다.

3.1.10 리뷰 보기

- 상단에는 영화의 정보(포스터, 제목, 감독, 개봉년도)가, 하단에는 리뷰 내용(한줄 평, 평점, 명장면)이 출력된다.
- 하단에는 리뷰 수정/삭제 버튼이 있다.
- (설계서에서 있었던 목록 버튼은 없었는데, 페이지마다 메뉴가 있기 때문에 리스트로 이동하는 것은 쉬울 것이라 생각하여 삭제했다.)
- 해당 리뷰에 댓글이 있다면 댓글도 작성일 오름차순으로 정렬되어 출력된다.

3.1.11 리뷰 댓글 등록

- 3.1.10의 [리뷰 보기]의 하단에 댓글 목록이 나오며 그 밑에서 댓글을 등록할 수 있다.
- 리뷰 댓글을 남기기 위해서는 닉네임과 비밀번호를 입력하고 댓글 내용을 작성할 수 있다.
- 비밀번호는 수정/삭제에 이용되고 수정할 수 없으므로 유의해야 한다.

3.1.12 리뷰 댓글 수정

- 3.1.10의 [리뷰 보기]의 하단에 댓글 목록이 나오는데, 테이블의 마지막 열에 있는 수정 버튼을 눌러 수정할 수 있다.
- 수정 시 두 개의 알림창이 뜨는데, 차례대로 수정할 내용과 비밀번호를 입력한다.
- 비밀번호가 댓글 작성 시 등록한 비밀번호와 일치하면 수정된다.

3.1.13 리뷰 댓글 삭제

- (설계서에는 3.1.12의 [리뷰 댓글 수정] 부분과 use case를 함께 작성했는데, 관리자의 비밀번호로도 삭제할 수 있게 하였고 기능이 다르므로 분리하였다.)
- 3.1.10의 [리뷰 보기]의 하단에 댓글 목록이 나오는데, 테이블의 마지막 열에 있는 삭제 버튼을 눌러 삭제할 수 있다.
- 삭제 시 비밀번호를 입력하는 알림창이 뜬다.
- 비밀번호가 댓글 작성 시 등록한 비밀번호와 일치하거나 관리자의 비밀번호와 일치하면 수정한다.

3.1.14 리뷰 댓글 검색

- (설계서에는 없지만 추가된 내용이다.)
- 3.1.10의 [리뷰 보기] 화면에서 댓글 목록, 댓글 작성 칸 밑에 댓글 검색 기능이 있다.
- 댓글 작성자로 검색하여 특정 닉네임으로 작성된 댓글을 검색 가능하다.

3.2 UI Design

3.2.1 관리자 영화 등록

HORROR MOVIE REVIEW BLOG

리뷰 목록 보기

리뷰 안 된 영화 목록 보기

전체 영화 목록 보기

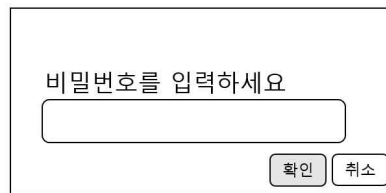
영화 등록하기

영화 등록

영화 제목	<input type="text" value="제목 입력"/>
개봉년도	<input type="text" value="년도 입력"/>
감독	<input type="text" value="감독 입력"/>
포스터	<input type="text" value="이미지 주소 입력"/>
	<input type="button" value="등록"/>

[그림 24] 영화 등록 화면 설계

3.2.2 관리자 영화 삭제



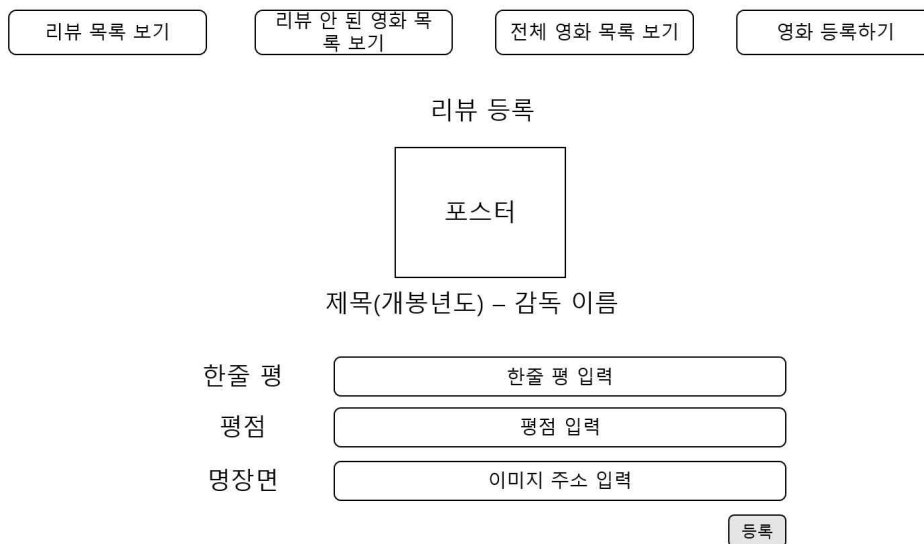
비밀번호를 입력하세요

[그림 25] 영화 삭제 화면 설계

- 영화 삭제 버튼은 [3.2.8]의 전체 영화 리스트 보기에서 누를 수 있다.

3.2.3 관리자 리뷰 등록

HORROR MOVIE REVIEW BLOG



리뷰 목록 보기 리뷰 안 된 영화 목록 보기 전체 영화 목록 보기 영화 등록하기

리뷰 등록

포스터

제목(개봉년도) - 감독 이름

한줄 평

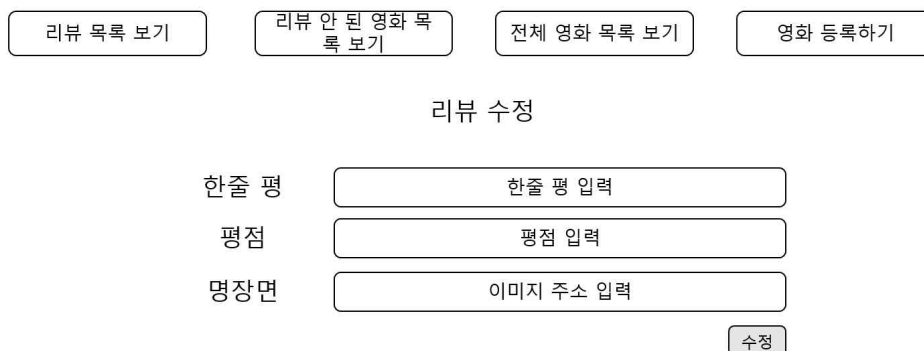
평점

명장면

[그림 26] 리뷰 등록 화면 설계

3.2.4 관리자 리뷰 수정

HORROR MOVIE REVIEW BLOG



리뷰 목록 보기 리뷰 안 된 영화 목록 보기 전체 영화 목록 보기 영화 등록하기

리뷰 수정

한줄 평

평점

명장면

[그림 27] 리뷰 수정 화면 설계

3.2.5 관리자 리뷰 삭제

- [그림 25]와 일치한다.
- 리뷰 삭제 버튼은 [3.2.10]의 리뷰 보기에서 누를 수 있다.

3.2.6 리뷰 리스트 보기

HORROR MOVIE REVIEW BLOG

[리뷰 목록 보기](#)[리뷰 안 된 영화 목록 보기](#)[전체 영화 목록 보기](#)[영화 등록하기](#)

리뷰 목록

순번	포스터	제목(개봉년도)	감독	명장면
1	포스터1	제목1(2022)	감독1	명장면1
	한줄평: 한줄평1 (4.5)			
2	포스터2	제목2(2018)	감독2	명장면2
	한줄평: 한줄평2 (4.0)			
3	포스터3	제목3(2021)	감독3	명장면3
	한줄평: 한줄평3 (1.9)			

[그림 28] 리뷰 리스트 보기 화면 설계

3.2.7 리뷰 안 된 영화 리스트 보기

HORROR MOVIE REVIEW BLOG

[리뷰 목록 보기](#)[리뷰 안 된 영화 목록 보기](#)[전체 영화 목록 보기](#)[영화 등록하기](#)

리뷰 안 된 영화 목록

순번	제목(개봉년도)	감독	포스터	등록/삭제	
1	제목1(2022)	감독1	포스터1	등록	삭제
2	제목2(2018)	감독2	포스터2	등록	삭제
3	제목3(2021)	감독3	포스터3	등록	삭제

[그림 29] 리뷰 안 된 영화 리스트 보기 화면 설계

3.2.8 전체 영화 리스트 보기

HORROR MOVIE REVIEW BLOG

[리뷰 목록 보기](#)[리뷰 안 된 영화 목록 보기](#)[전체 영화 목록 보기](#)[영화 등록하기](#)

전체 영화 목록

순번	제목(개봉년도)	감독	포스터	등록
1	제목1(2022)	감독1	포스터1	등록
2	제목2(2018)	감독2	포스터2	등록
3	제목3(2021)	감독3	포스터3	등록

제목 검색

[검색](#)

[그림 30] 전체 영화 리스트 보기 화면 설계

3.2.9 영화 검색

- [그림 30]와 동일하다.

3.2.10 리뷰 보기

HORROR MOVIE REVIEW BLOG

[리뷰 목록 보기](#)[리뷰 안 된 영화 목록 보기](#)[전체 영화 목록 보기](#)[영화 등록하기](#)

리뷰 보기

제목(개봉년도)	감독	포스터
제목1(2022)	감독1	포스터1
한줄 평	한줄 평1	
평점	4.5	
명장면	명장면1	

[수정](#)[삭제](#)

닉네임1	댓글1	수정 삭제
닉네임2	댓글2	수정 삭제
닉네임 입력 ●●●●●	댓글 입력	등록

작성자로 댓글 검색

[검색](#)

[그림 31] 리뷰 보기 화면 설계

3.2.11 리뷰 댓글 등록

- [그림 31]와 동일하다.

3.2.12 리뷰 댓글 수정

- [그림 31]와 동일하다.

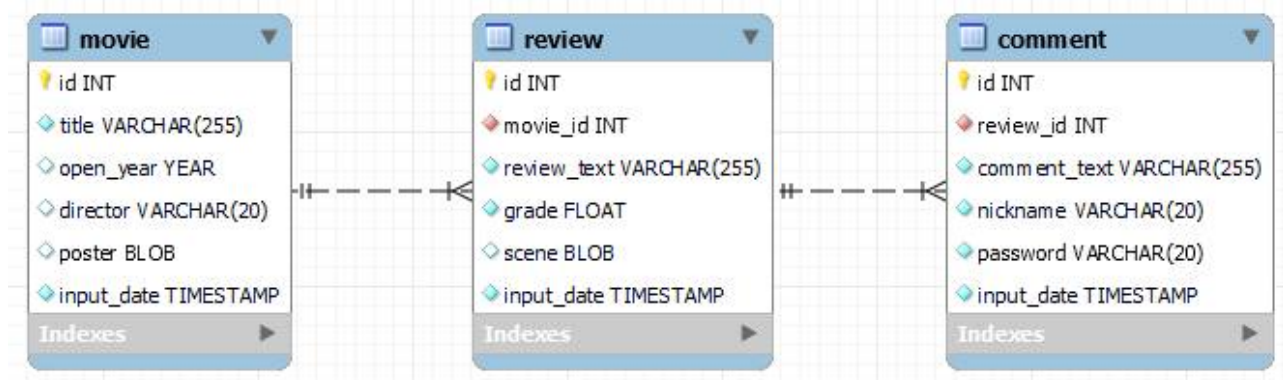
3.2.13 리뷰 댓글 삭제

- [그림 31]와 동일하다.

3.2.14 리뷰 댓글 검색

- [그림 31]와 동일하다.

3.3 Conceptual Design



[그림 32] E-R 다이어그램

3.4 Logical Design

movie
<pre>CREATE TABLE movie(id INT NOT NULL AUTO_INCREMENT, title VARCHAR(255) NOT NULL, open_year YEAR, director VARCHAR(20), poster BLOB, input_date TIMESTAMP NOT NULL, PRIMARY KEY(id)) ENGINE=InnoDB DEFAULT CHARACTER SET utf8 DEFAULT COLLATE utf8_general_ci;</pre>

	Field	Type	Null	Key	Default	Extra
►	id	int	NO	PRI	NULL	auto_increment
	title	varchar(255)	NO		NULL	
	open_year	year	YES		NULL	
	director	varchar(20)	YES		NULL	
	poster	blob	YES		NULL	
	input_date	timestamp	NO		NULL	

review

```
CREATE TABLE review(
    id INT NOT NULL AUTO_INCREMENT,
    movie_id INT NOT NULL,
    review_text VARCHAR(255) NOT NULL,
    grade FLOAT NOT NULL,
    scene BLOB,
    input_date TIMESTAMP NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(movie_id) REFERENCES movie(id)
    ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE=InnoDB
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
```

	Field	Type	Null	Key	Default	Extra
►	id	int	NO	PRI	NULL	auto_increment
	movie_id	int	NO	MUL	NULL	
	review_text	varchar(255)	NO		NULL	
	grade	float	NO		NULL	
	scene	blob	YES		NULL	
	input_date	timestamp	NO		NULL	

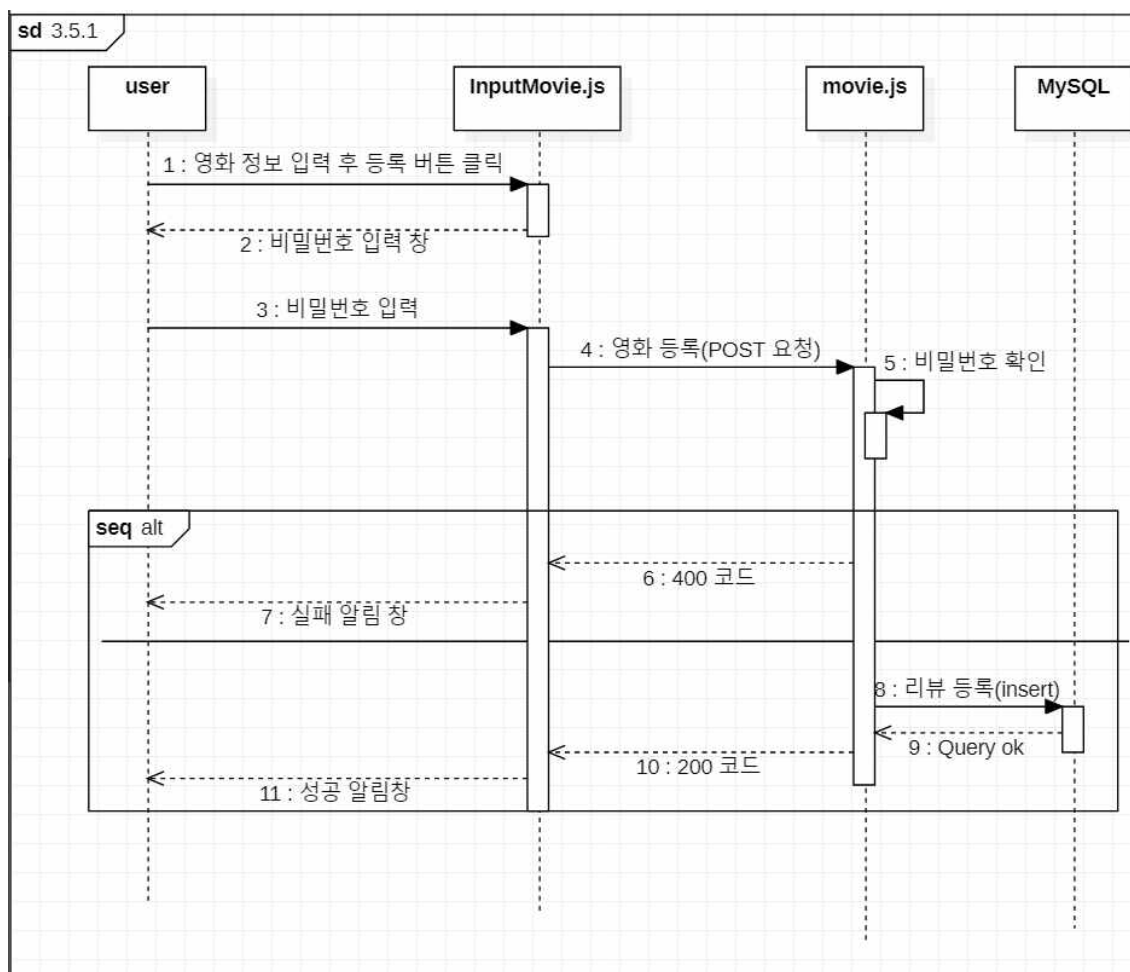
comment

```
CREATE TABLE comment(
    id INT NOT NULL AUTO_INCREMENT,
    review_id INT NOT NULL,
    comment_text varchar(255) not null,
    nickname varchar(20) not null,
    password varchar(20) not null,
    input_date TIMESTAMP NOT NULL,
    PRIMARY KEY(id),
    FOREIGN KEY(review_id) REFERENCES review(id)
    ON UPDATE CASCADE ON DELETE CASCADE
) ENGINE=InnoDB
DEFAULT CHARACTER SET utf8
DEFAULT COLLATE utf8_general_ci;
```


	Field	Type	Null	Key	Default	Extra
▶	id	int	NO	PRI	NULL	auto_increment
	review_id	int	NO	MUL	NULL	
	comment_text	varchar(255)	NO		NULL	
	nickname	varchar(20)	NO		NULL	
	password	varchar(20)	NO		NULL	
	input_date	timestamp	NO		NULL	

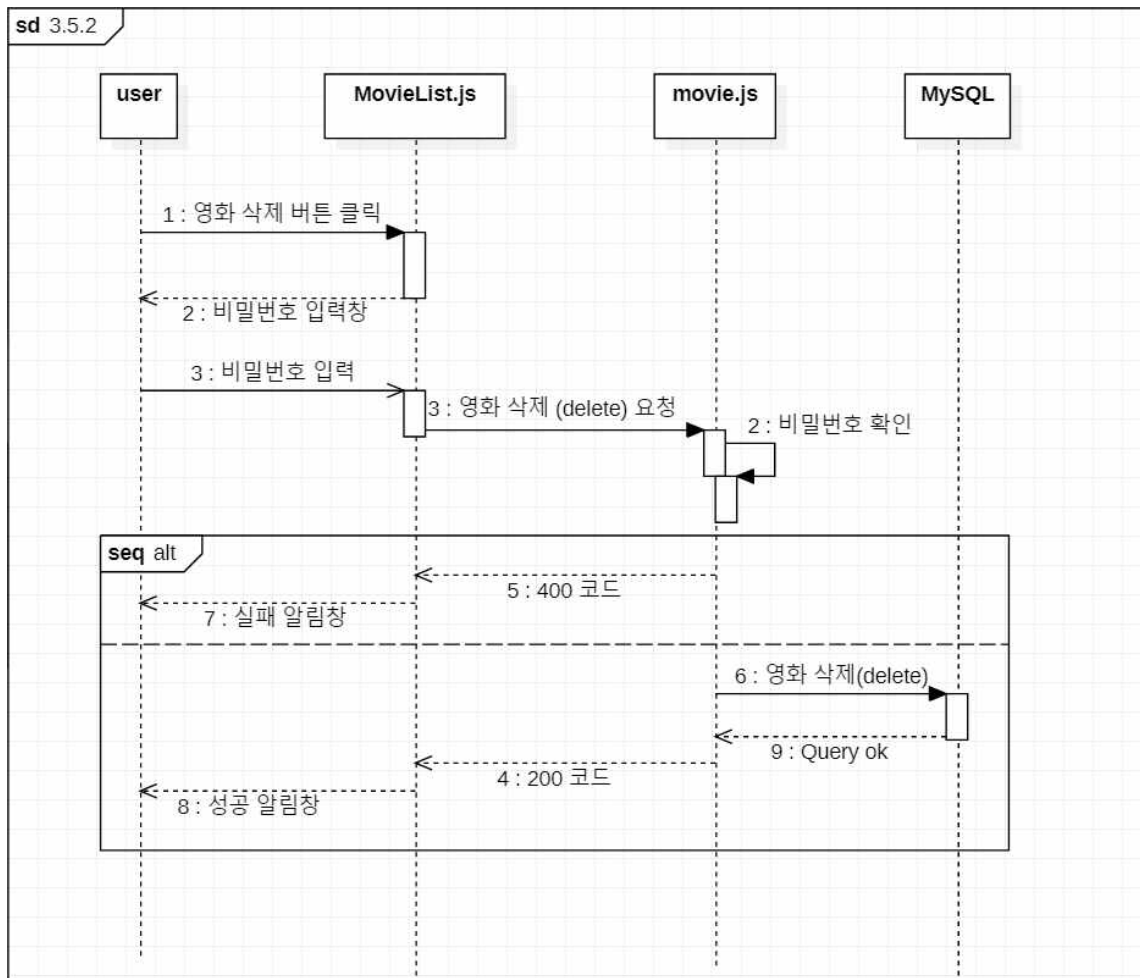
3.5 Sequence Diagram

3.5.1 관리자 영화 등록



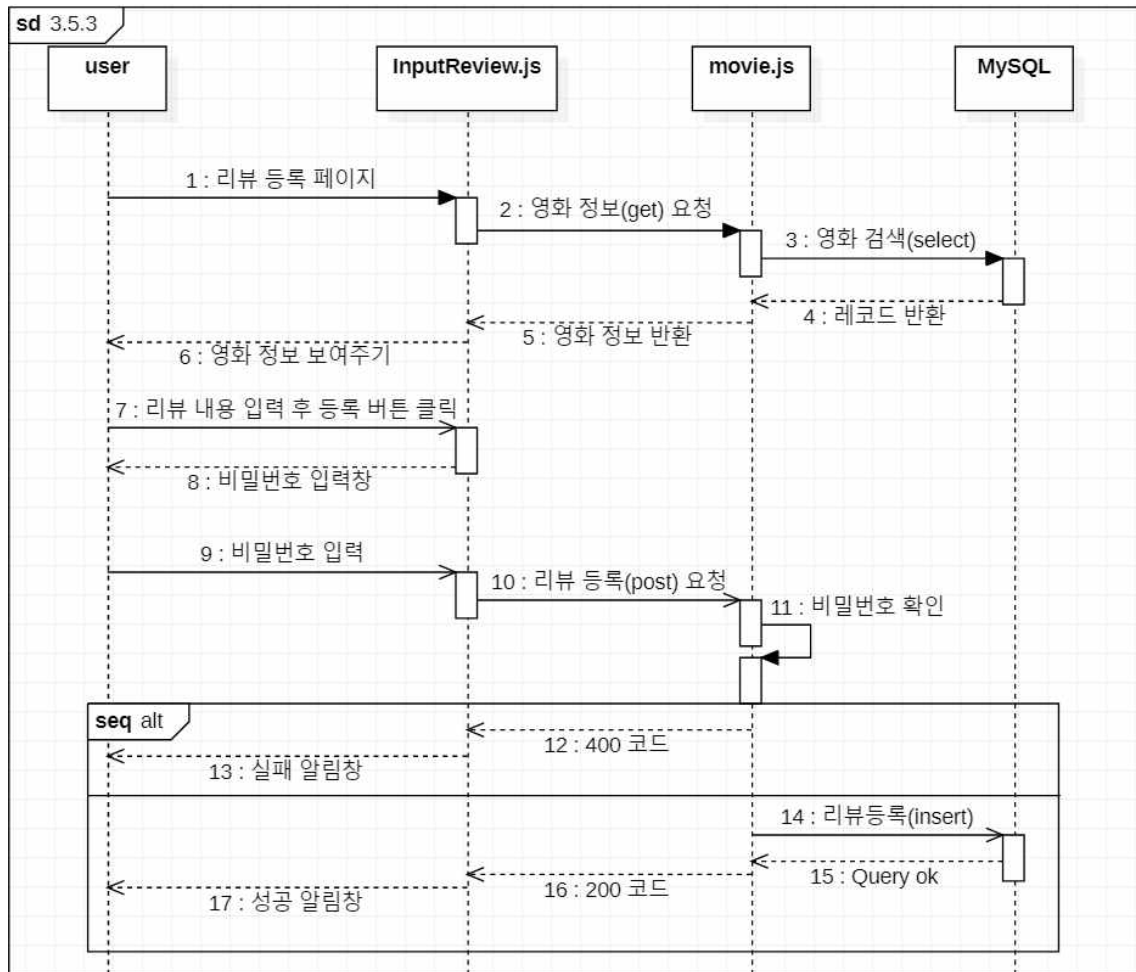
[그림 33] 관리자 영화 등록 시퀀스 다이어그램

3.5.2 관리자 영화 삭제



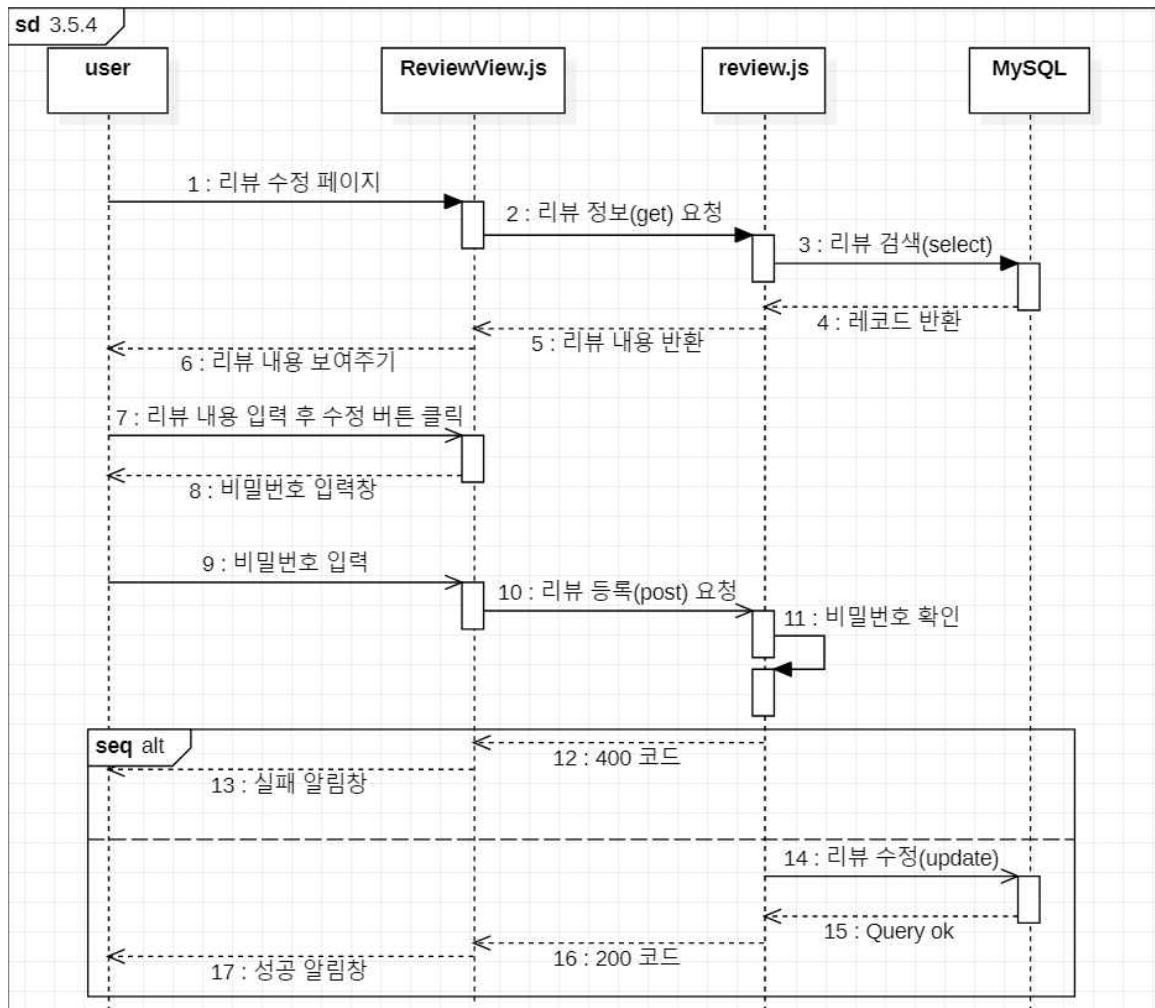
[그림 34] 관리자 영화 삭제 시퀀스 다이어그램

3.5.3 관리자 리뷰 등록



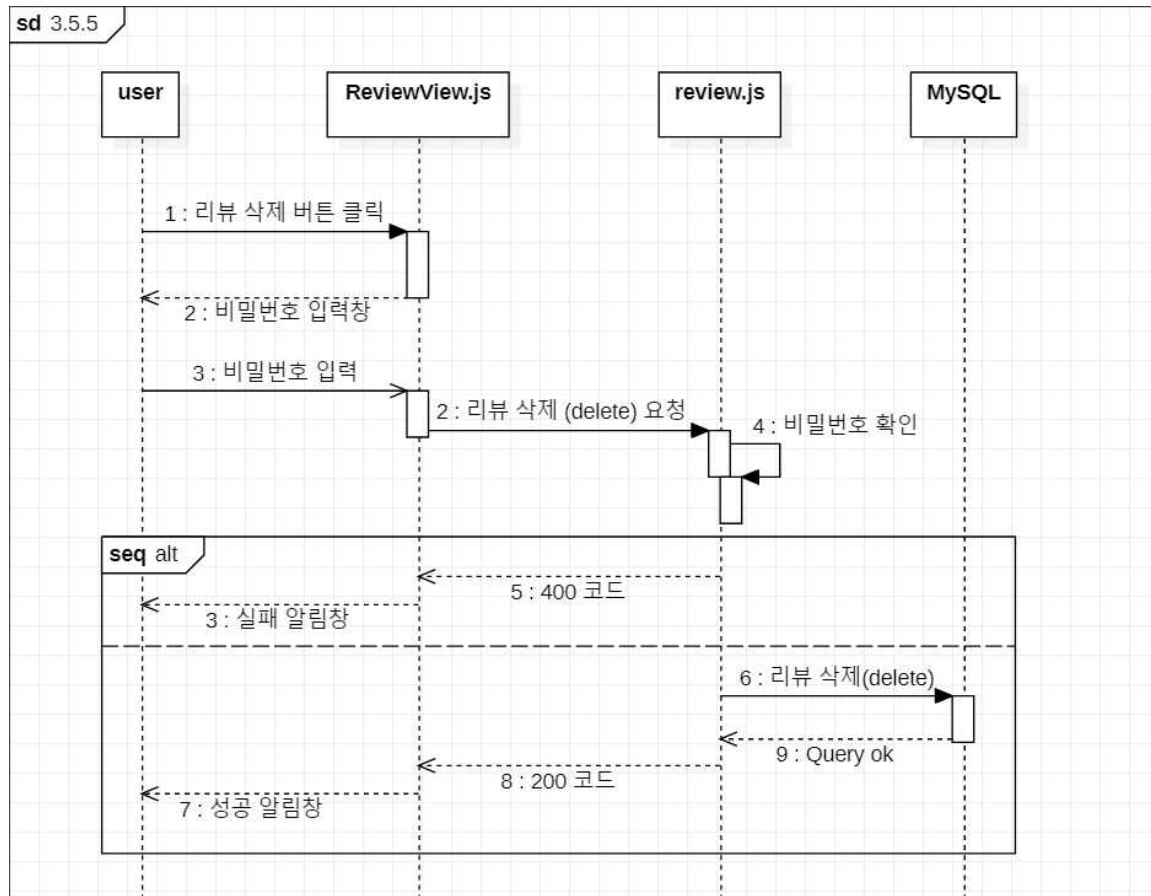
[그림 35] 관리자 리뷰 등록 시퀀스 다이어그램

3.5.4 관리자 리뷰 수정



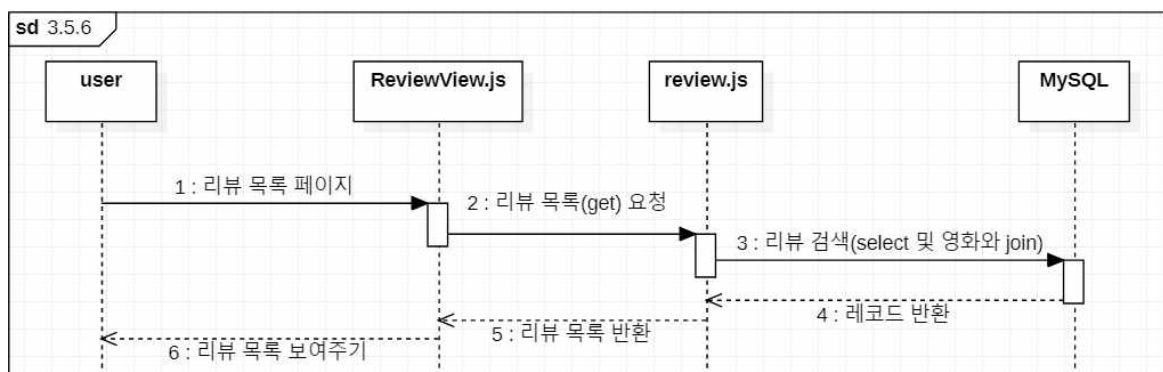
[그림 36] 관리자 리뷰 수정 시퀀스 다이어그램

3.5.5 관리자 리뷰 삭제



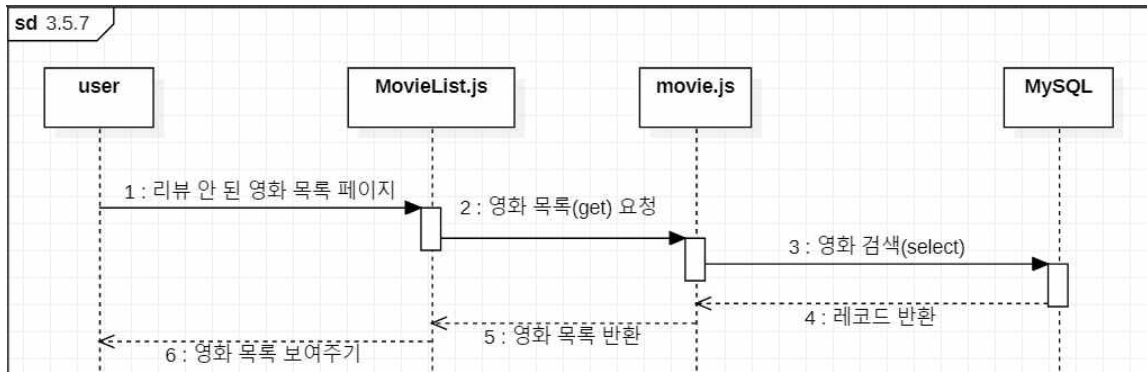
[그림 37] 관리자 리뷰 삭제 시퀀스 다이어그램

3.5.6 리뷰 리스트 보기



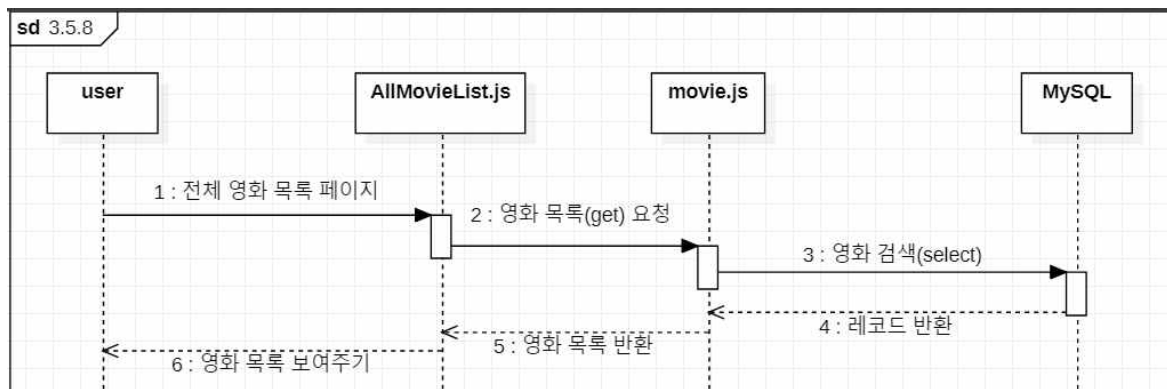
[그림 38] 리뷰 리스트 보기 시퀀스 다이어그램

3.5.7 리뷰 안 된 영화 리스트 보기



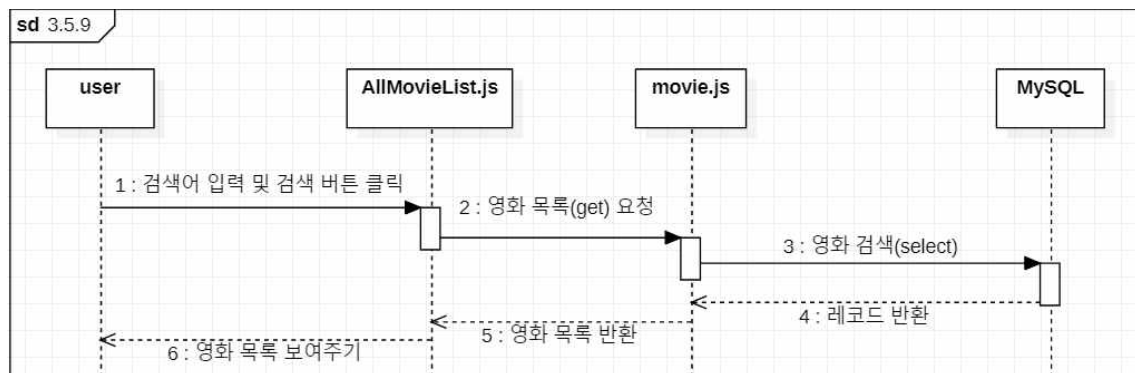
[그림 39] 리뷰 안 된 영화 리스트 보기 시퀀스 다이어그램

3.5.8 전체 영화 리스트 보기



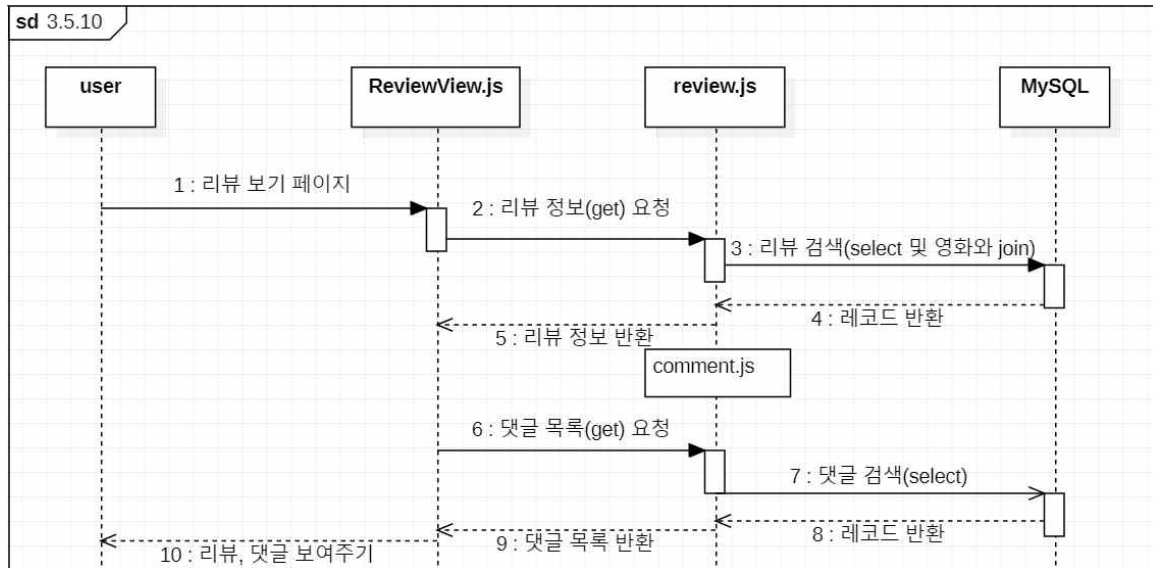
[그림 40] 전체 영화 리스트 보기 시퀀스 다이어그램

3.5.9 영화 검색



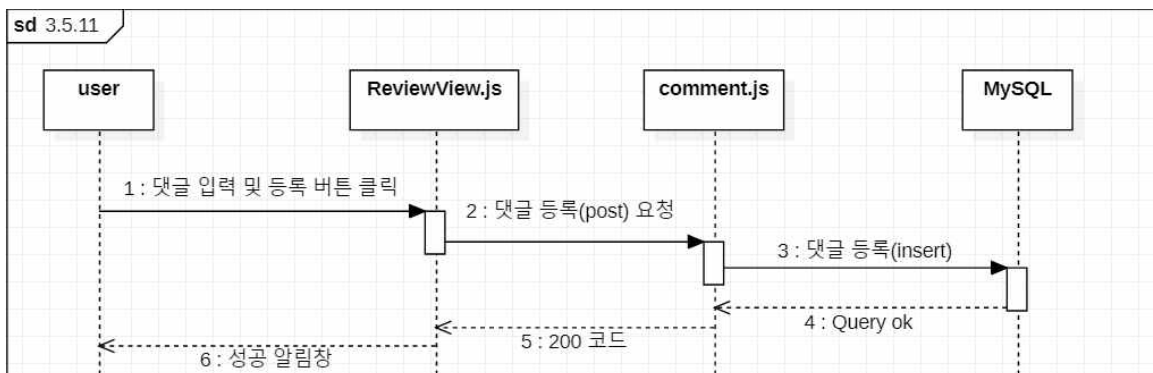
[그림 41] 영화 검색 시퀀스 다이어그램

3.5.10 리뷰 보기



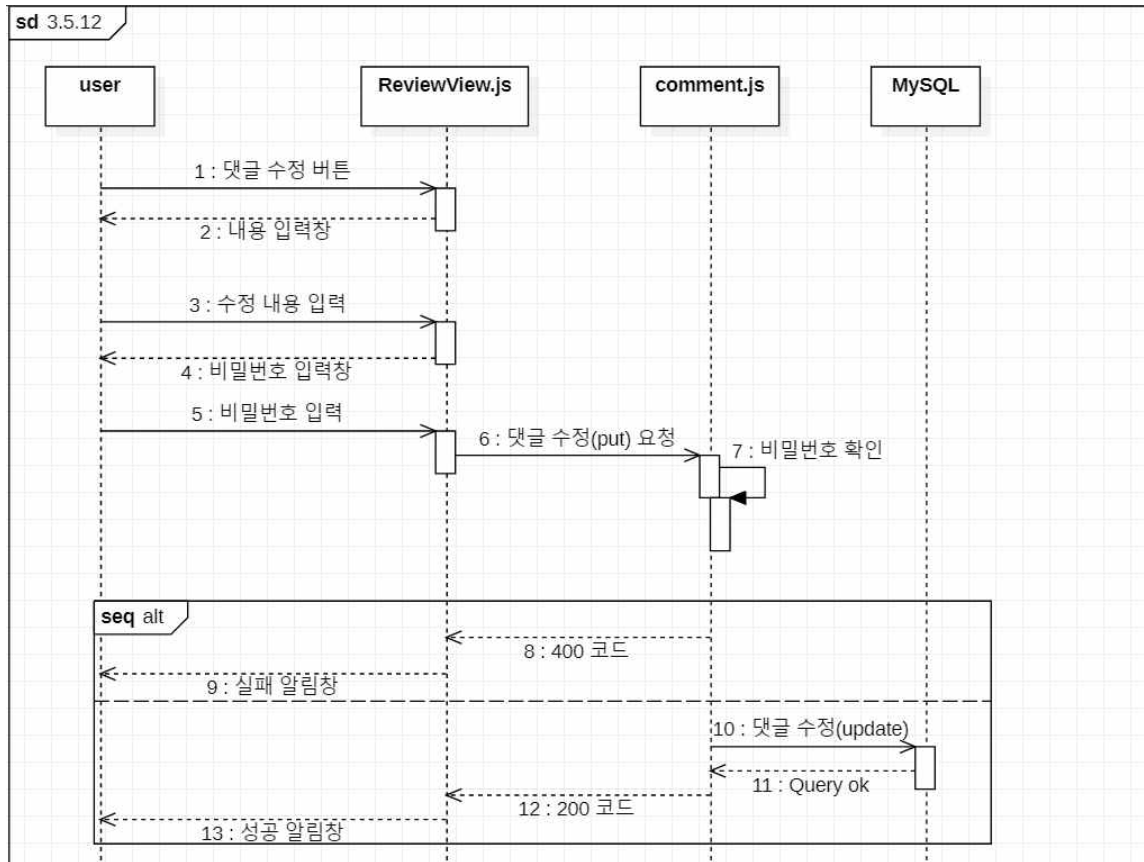
[그림 42] 리뷰 보기 시퀀스 다이어그램

3.5.11 리뷰 댓글 등록



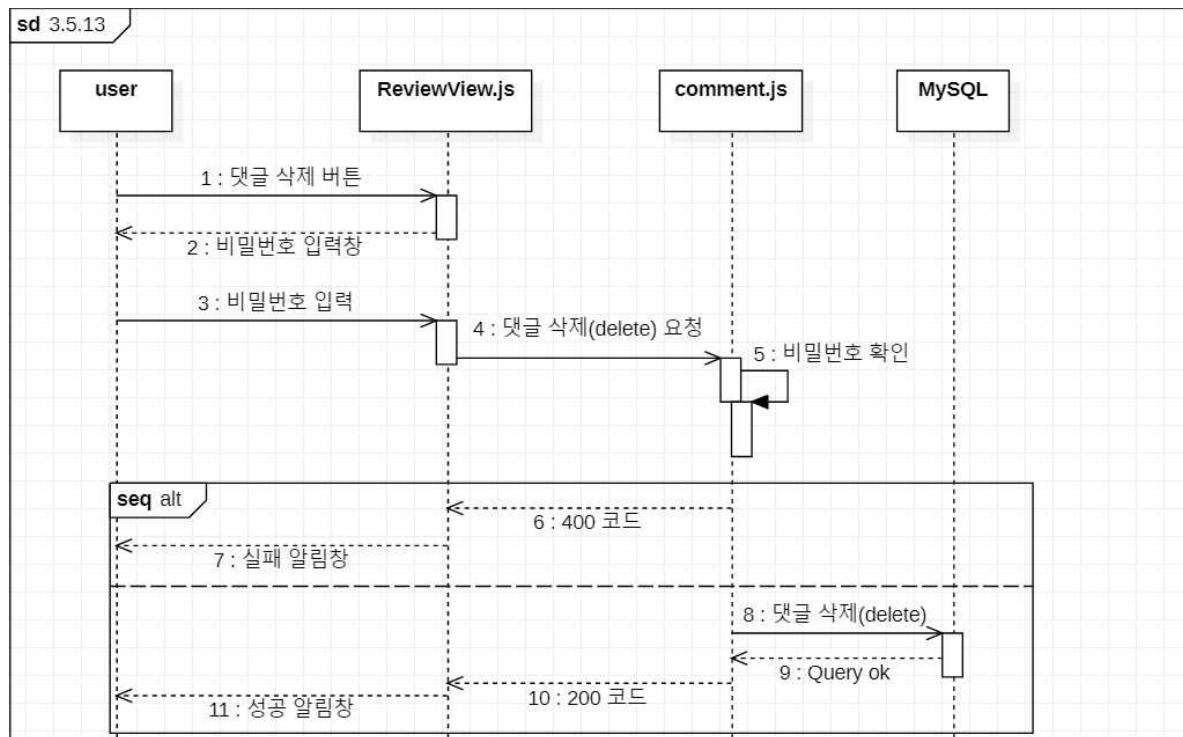
[그림 43] 리뷰 댓글 등록 시퀀스 다이어그램

3.5.12 리뷰 댓글 수정



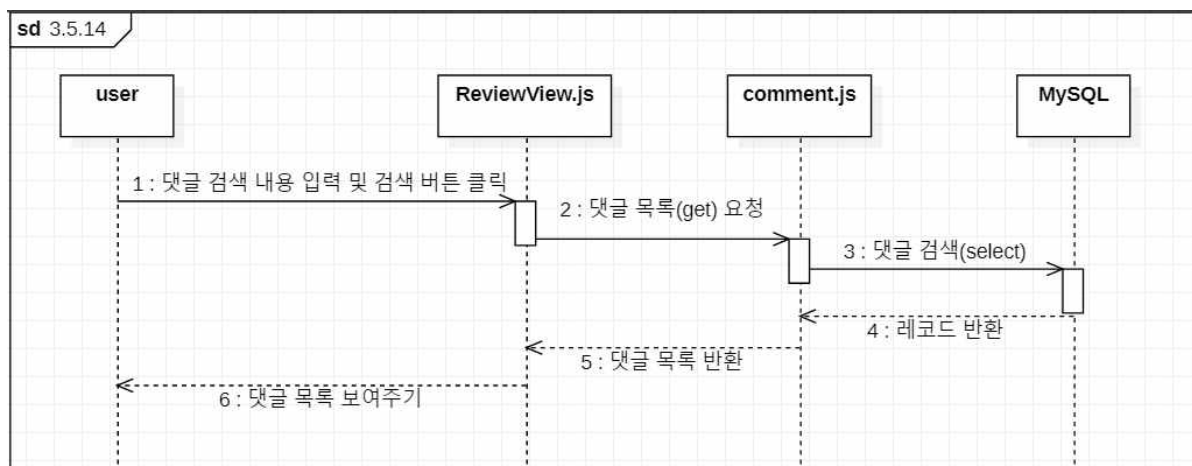
[그림 44] 리뷰 댓글 수정 시퀀스 다이어그램

3.5.13 리뷰 댓글 삭제



[그림 45] 리뷰 댓글 삭제 시퀀스 다이어그램

3.5.14 리뷰 댓글 검색



[그림 46] 리뷰 댓글 검색 시퀀스 다이어그램

3.6 자료 구조 설계

- 영화

```
{
  "array": [
    {
      "id": "영화 id",
      "director": "영화 감독",
      "input_date": "영화 등록 시간",
      "open_year": "개봉 년도",
      "poster": "영화 포스터 이미지 주소",
      "title": "영화 제목"
    },
    { ... }
  ]
}
```

- 리뷰

```
{
  "review": {
    "id": "리뷰 id",
    "movie_id": "영화 id",
    "input_date": "리뷰 등록 시간",
    "review_text": "한줄 평",
    "grade": "평점",
    "scene": "명장면 이미지 주소"
  }
}
```

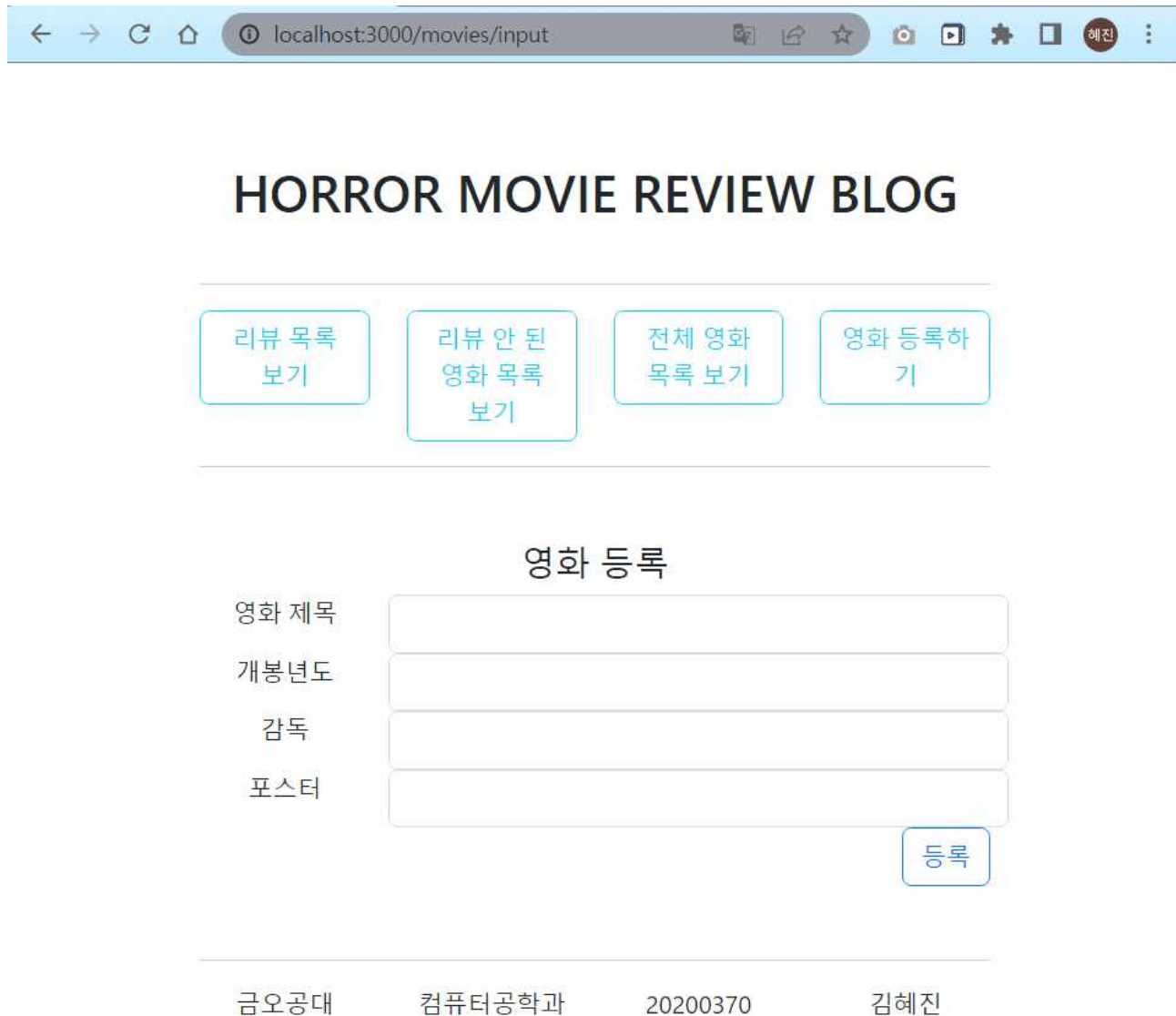
- 댓글

```
{
  "array": [
    {
      "id": "댓글 id",
      "review_id": "리뷰 id",
      "input_date": "댓글 등록 시간",
      "comment_text": "댓글 내용",
      "nickname": "작성자 닉네임"
    },
    { ... }
  ]
}
```

4. 구현

4.1.1 관리자 영화 등록

- 실행 화면



The screenshot shows a web browser at localhost:3000/movies/input. The page title is "HORROR MOVIE REVIEW BLOG". Below the title are four buttons: "리뷰 목록 보기", "리뷰 안 된 영화 목록 보기", "전체 영화 목록 보기", and "영화 등록하기". The "영화 등록하기" button is highlighted. Below these buttons is a form titled "영화 등록" with four input fields: "영화 제목", "개봉년도", "감독", and "포스터". A "등록" button is at the bottom right of the form. At the bottom of the page, there is a footer with the text: "금오공대 컴퓨터공학과 20200370 김혜진".

localhost:3000/movies/input

HORROR MOVIE REVIEW BLOG

리뷰 목록 보기

리뷰 안 된 영화 목록 보기

전체 영화 목록 보기

영화 등록하기

영화 등록

영화 제목

개봉년도

감독

포스터

등록

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 영화 등록 초기 상태 화면

HORROR MOVIE REVIEW BLOG

영화 등록하
기

영하 17도

1/10

김혜진

[그림] 영화 등록 중간 상태(내용 작성) 화면

localhost:3000 내용:

비밀번호를 입력하세요

확인

취소

리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

영화 등록

영화 제목

인시디어스

개봉년도

2010

감독

제임스 완

포스터

https://movie-phinf.pstatic.net/20120827_268/134604

등록

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 영화 등록 중간 상태(비밀번호 입력) 화면

HORROR MOVIE REVIEW BLOG


리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

리뷰 안 된 영화 목록

순 번	제목(개봉년도)	감독	포스터	등록/삭제
1	인시디어스 (2010)	제임스 완		<div>등록</div> <div>삭제</div>

금오공대 컴퓨터공학과 20200370 김혜진
[그림] 영화 등록 최종 상태(리뷰 안 된 영화 목록으로 이동) 화면

- 소스 코드

```

InputMovie.js
// 20200370 김혜진
const { useState, useEffect } = require('react');
const axios = require('axios');

function InputMovie() {
  const [title, setTitle] = useState('');
  const [year, setYear] = useState('');
  const [director, setDirector] = useState('');
  const [poster, setPoster] = useState('');

  /* 입력된 정보 핸들링 */
  const handleTitle = (e) => {
    setTitle(e.target.value)
  }

```

```

const handleYear = (e) => {
  setYear(e.target.value)
}
const handleDirector = (e) => {
  setDirector(e.target.value)
}
const handlePoster = (e) => {
  setPoster(e.target.value)
}

const handleClickInput = () => { // 등록 버튼 클릭이벤트 핸들러
  let password = prompt('비밀번호를 입력하세요');
  let obj = {
    director: director,
    title: title,
    year: year,
    poster: poster
  };
  let config = {
    headers: { Authorization: password }
  };
  axios.post('http://localhost:3030/movies', obj, config) // 영화 정보와 비밀번호로 /movies
  에 post 요청
  .then(res => {
    alert("성공!");
    window.location.href="/movies"; // 비밀번호 일치 및 DB에 삽입 성공 시 영화 목록으로 이
    동
  })
  .catch(err => {
    alert("실패!");
  })
}

return (
  <div class="container my-5">
    <div class="row">
      <div class="col h4">
        영화 등록
      </div>
    </div>
    <div class="row">
      <label for="movie_title" class="form-label col-3">영화 제목</label>
      <input
        class="form-control col"
        id="movie_title"
        value = {title}
        onChange = {handleTitle}
      />
    </div>
    <div class="row">
      <label for="movie_openYear" class="form-label col-3">개봉년도</label>
      <input

```

```

        class="form-control col"
        id="movie_openYear"
        value = {year}
        onChange = {handleYear}
      />
    </div>
    <div class="row">
      <label for="movie_director" class="form-label col-3">감독</label>
      <input
        class="form-control col"
        id="movie_director"
        value = {director}
        onChange = {handleDirector}
      />
    </div>
    <div class="row">
      <label for="movie_poster" class="form-label col-3">포스터</label>
      <input
        class="form-control col"
        id="movie_poster"
        value = {poster}
        onChange = {handlePoster}
      />
    </div>

    <div class="row">
      <div class="col text-end">
        <button class="btn btn-outline-primary" onClick={handleClickInput}>등록</button>
      </div>
    </div>
  </div>
);
}

```

export default InputMovie;

movie.js의 일부

// 20200370 김혜진

```

router.post('/', async (req, res) => {
  if (adminPW !== req.headers.authorization) { // 비밀번호 확인(관리자만)
    res.status(400).send();
    return;
  }

  let movieObj = {
    title: req.body.title,
    open_year: req.body.year,
    director: req.body.director,
    poster: req.body.poster,
    input_date: new Date()
  };

  if (await SendQuery("INSERT INTO movie SET ?", movieObj)) // INSERT 요청

```



```

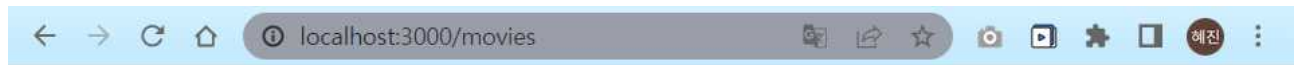
res.status(200).send();
else
res.status(400).send();
})

```

- 관리자 영화 등록은 관리자가 폼에 입력한 바탕으로 INSERT로 쿼리를 보내 진행된다.
- 성공 시 200 코드를 받으면 성공 창이 뜨고 리뷰 안 된 영화 목록으로 이동한다.
- 관리자 비밀번호가 다르거나 DB에 INSERT 실패 시 400 코드를 반환하며 실패 창이 뜬다.

4.1.2 관리자 영화 삭제

- 실행 화면



HORROR MOVIE REVIEW BLOG


리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

리뷰 안 된 영화 목록

순 번	제목(개봉년도)	감독	포스터	등록/삭제
1	인시디어스 (2010)	제임스 완		<div>등록</div> <div>삭제</div>

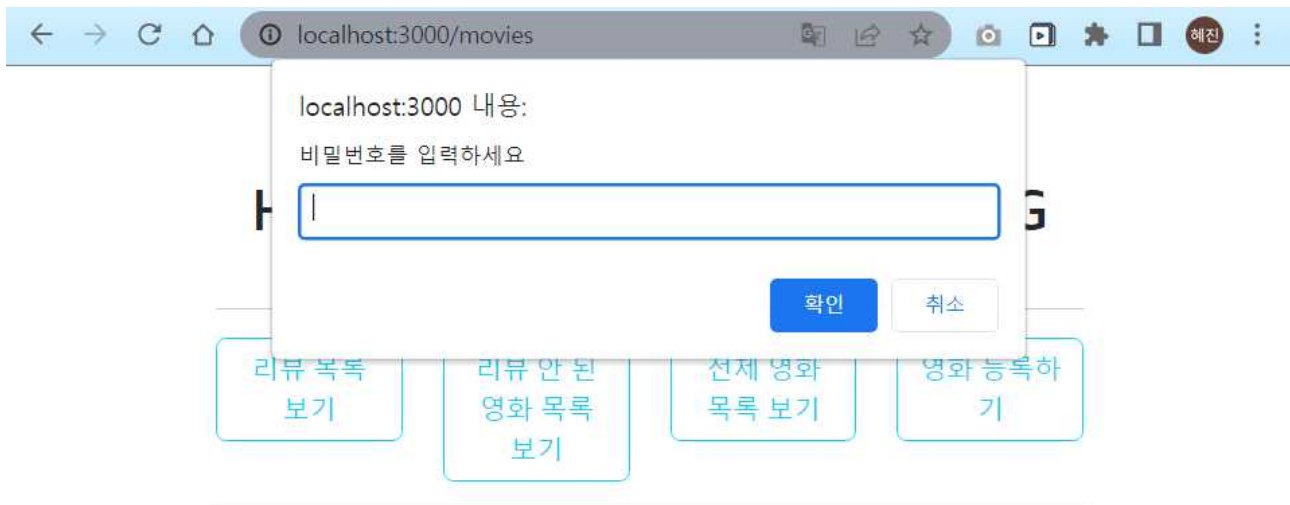
금오공대

컴퓨터공학과

20200370

김혜진

[그림] 영화 삭제 초기 상태 화면



리뷰 안 된 영화 목록

순 번	제목(개봉년도)	감독	포스터	등록/삭제
1	인시디어스 (2010)	제임스 완		<div>등록</div> <div>삭제</div>

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 영화 삭제 중간 상태(비밀번호 입력) 화면

HORROR MOVIE REVIEW BLOG

리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

리뷰 안 된 영화 목록

순번	제목(개봉년도)	감독	포스터	등록/삭제
금오공대	컴퓨터공학과	20200370	김혜진	

[그림] 영화 삭제 최종 상태(리뷰 안 된 영화 목록으로 이동) 화면

- 소스 코드

```
MovieList.js의 일부
// 20200370 김혜진
import { useState, useEffect } from 'react'
import axios from 'axios'

const Movie = (prop) => {

  const handleClickDelete = (e) => { // 삭제 버튼 클릭이벤트 핸들러
    let password = prompt('비밀번호를 입력하세요');
    let config = {
      headers: { Authorization: password }
    };
    axios.delete('http://localhost:3030/movies/'+e.target.name, config) // /movie/:id로 비밀번호
```

호를 담아 delete 요청

```
.then(res => {  
  console.log(res);  
  alert("성공!");  
  window.location.href="/movies"; // 성공 시 /movies로 이동하여 새로고침  
})  
.catch(err => {  
  alert("실패!");  
  console.log(err);  
})  
}
```

movie.js의 일부

// 20200370 김혜진

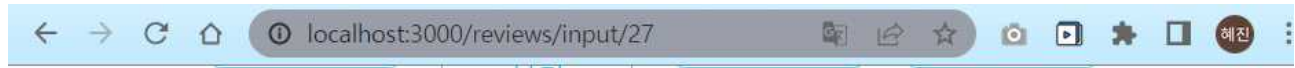
```
router.delete('/:id', async (req, res) => {  
  if (adminPW !== req.headers.authorization) { // 비밀번호 확인(관리자만)  
    res.status(400).send();  
    return;  
  }  
  
  let result = await SendQuery("DELETE FROM movie where id=?", req.params.id); // DELETE 요청  
  if (result !== null) {  
    console.log(result);  
    res.status(200).send();  
  }  
  else {  
    res.status(400).send();  
  }  
})
```

• 삭제 비밀번호가 일치하면 영화 id가 조건인 DELETE 쿼리를 보내 삭제한다. 성공 시 200 코드를 보내고 성공 알림 창을 띄운 후 새로고침 한다.

• 비밀번호가 일치하지 않거나 DELETE 쿼리 응답이 실패 시 400 코드를 보내고 실패 알림 창을 띄운다.

4.1.3 관리자 리뷰 등록

- 실행 화면



보기

리뷰 등록



인시디어스(2010) - 제임스 완

한줄 평

평점

명장면

등록

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 등록 초기 상태 화면

리뷰 등록



인시디어스(2010) - 제임스 완

한줄 평	<input type="text" value="어릴 때는 무서웠는데 지금 다시 보니까 시시하다."/>
평점	<input type="text" value="3.0"/>
명장면	<input type="text" value="https://movie-phinf.pstatic.net/20120912_223/134743"/>

등록

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 등록 중간 상태(리뷰 내용 입력) 화면

localhost:3000 내용:

비밀번호를 입력하세요

확인

취소



인시디어스(2010) - 제임스 완

한줄 평

어릴 때는 무서웠는데 지금 다시 보니까 시시하다.

평점

3.0

명장면

https://movie-phinf.pstatic.net/20120912_223/134743

등록

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 등록 중간 상태(비밀번호 입력) 화면

HORROR MOVIE REVIEW BLOG

리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

리뷰 목록

순 번	포스터	제목(개 봉년도)	감 독	명장면
1		인시디어 스(2010)	제 임 스 완	
한줄평: 어릴 때는 무서웠는데 지금 다시 보니까 시시하다. (3)				

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 등록 최종 상태(리뷰 목록으로 이동) 화면

- 소스 코드

```

InputReview.js
// 20200370 김혜진
import { useParams } from 'react-router-dom';
const { useState, useEffect } = require('react');
const axios = require('axios');

function InputReview() {
  let [movie, setMovie] = useState({})
  useEffect(() => {
    getMovie()
  }, {})

```



```

let movieId = useParams().movieId;

const getMovie = async() => { // 영화 정보 받아오기
  try {
    const res = await axios.get('http://localhost:3030/movies/'+movieId); // /movies/:id에
    GET요청으로 영화 정보 받아오기
    setMovie(res.data[0]);
  } catch (e) {
    console.log(e);
  }
};

const [reviewText, setReviewText] = useState('');
const [grade, setGrade] = useState('');
const [scene, setScene] = useState('');

/* 입력 내용 핸들러 */
const handleReviewText = (e) => {
  setReviewText(e.target.value)
}
const handleGrade = (e) => {
  setGrade(e.target.value)
}
const handleScene = (e) => {
  setScene(e.target.value)
}

const handleClickInput = () => { // 등록 버튼 클릭이벤트 핸들러
  let password = prompt('비밀번호를 입력하세요');
  let obj = {
    movie_id: movieId,
    review_text: reviewText,
    grade: grade,
    scene: scene
  };
  let config = {
    headers: { Authorization: password }
  };
  axios.post('http://localhost:3030/reviews', obj, config) // /reviews에 post로 내용과 비밀
  번호를 담아 요청
  .then(res => {
    alert("성공!");
    window.location.href="/reviews"; // 성공 시 리뷰 목록으로 이동
  })
  .catch(err => {
    alert("실패!");
  })
}

return (
  <div class="container my-5">
    <div class="row">

```

```

    <div class="col h4">
      리뷰 등록
    </div>
  </div>
  <div class="row my-3">
    <div class="col">
      <img src={movie.poster}/>
    </div>
    <div class="col-12 my-2">
      <b>{movie.title}</b>({movie.open_year}) - {movie.director}
    </div>
  </div>
  <div class="row">
    <label for="reviewText" class="form-label col-3">한줄 평</label>
    <input
      class="form-control col"
      id="reviewText"
      value = {reviewText}
      onChange = {handleReviewText}
    />
  </div>
  <div class="row">
    <label for="movie_openYear" class="form-label col-3">평점</label>
    <input
      class="form-control col"
      id="grade"
      value = {grade}
      onChange = {handleGrade}
    />
  </div>
  <div class="row">
    <label for="scene" class="form-label col-3">명장면</label>
    <input
      class="form-control col"
      id="scene"
      value = {scene}
      onChange = {handleScene}
    />
  </div>

  <div class="row">
    <div class="col text-end">
      <button class="btn btn-outline-primary" onClick={handleClickInput}>등록</button>
    </div>
  </div>
</div>
);
}

export default InputReview;

```

review.js의 일부

// 20200370 김혜진

```

router.post('/', async (req, res) => {
  if (adminPW !== req.headers.authorization) { // 비밀번호 확인(관리자만)
    res.status(400).send();
    return;
  }

  let reviewObj = {
    movie_id: req.body.movie_id,
    review_text: req.body.review_text,
    grade: req.body.grade,
    scene: req.body.scene,
    input_date: new Date()
  };

  if (await SendQuery("INSERT INTO review SET ?;", reviewObj)) // review 테이블에 INSERT
  요청
    res.status(200).send();
  else
    res.status(400).send();
})

```

- 등록 시 최초에 get 요청과 영화 id를 주어 영화 정보를 받아온다.
- 관리자가 폼을 통해 입력한 정보를 받아 등록 버튼 이벤트가 일어나면 비밀번호를 입력받고 /review에 post로 요청한다.
- 서버에서는 INSERT 쿼리로 review 테이블에 등록을 요청하고 성공 시 200 코드를 보낸다.
- 비밀번호가 맞지 않거나 INSERT 실패 시 400 코드를 보내고, 프론트 측에서 실패 알림창을 띄운다.
- 200 코드를 받으면 성공 알림창을 띄우고 리뷰 목록으로 이동한다.

4.1.4 관리자 리뷰 수정

- 실행 화면

The screenshot shows a web browser at localhost:3000/reviews/update/9. The page title is 'HORROR MOVIE REVIEW BLOG'. Below the title are four buttons: '리뷰 목록 보기' (View Review List), '리뷰 안 된 영화 목록 보기' (View Not Reviewed Movie List), '전체 영화 목록 보기' (View All Movie List), and '영화 등록하기' (Register Movie). The main section is titled '리뷰 수정' (Edit Review). It contains three input fields: '한줄 평' (One-line Review) with the text '어릴 때는 무서웠는데 지금 다시 보니까 시시하다.' (I was scared when I was young, but now it's boring when I watch it again.), '평점' (Rating) with the value '3', and '명장면' (Highlight) with the URL 'https://movie-phinf.pstatic.net/20120912_223/134743'. A '수정' (Update) button is at the bottom right. The footer displays '금오공대' (Gyeongsang National University), '컴퓨터공학과' (Department of Computer Science), '20200370', and '김혜진' (Kim Hyejin).

localhost:3000/reviews/update/9

HORROR MOVIE REVIEW BLOG

리뷰 목록 보기 리뷰 안 된 영화 목록 보기 전체 영화 목록 보기 영화 등록하기

리뷰 수정

한줄 평 어릴 때는 무서웠는데 지금 다시 보니까 시시하다.

평점 3

명장면 https://movie-phinf.pstatic.net/20120912_223/134743

수정

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 수정 초기 상태 화면

HORROR MOVIE REVIEW BLOG

리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

리뷰 수정

한줄 평

어릴 때는 무서웠는데 지금 다시 보니까 시시하다.

평점

2

명장면

https://movie-phinf.pstatic.net/20120912_223/134743

수정

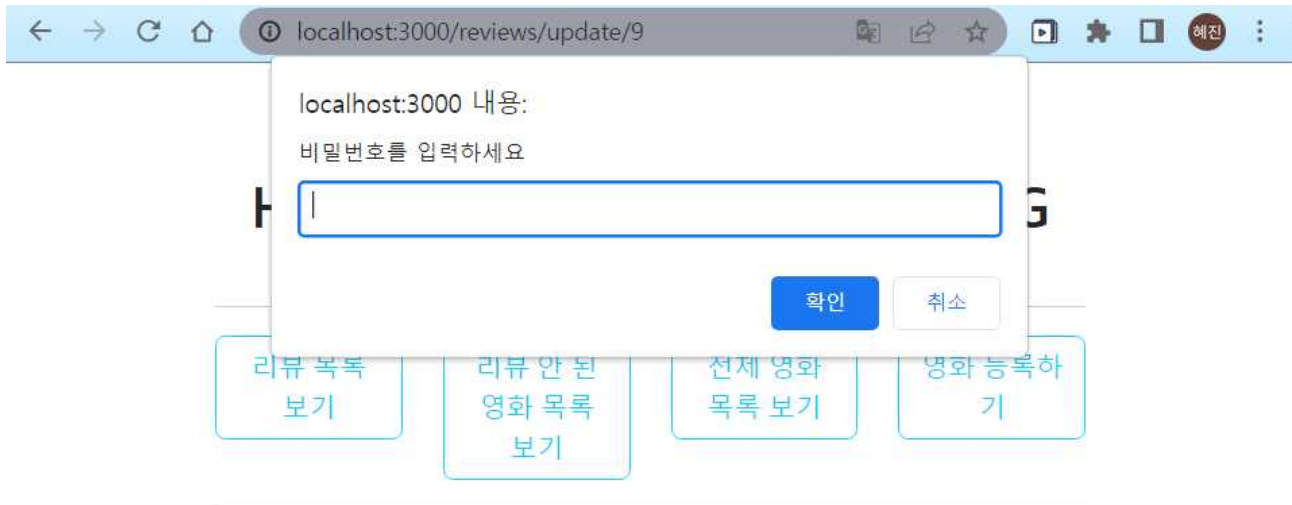
금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 수정 중간 상태(리뷰 내용 - 여기서는 평점 수정) 화면



리뷰 수정

한줄 평	어릴 때는 무서웠는데 지금 다시 보니까 시시하다.
평점	2
명장면	https://movie-phinf.pstatic.net/20120912_223/134743
<button>수정</button>	

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 수정 중간 상태(비밀번호 입력) 화면

리뷰 보기

제목(개봉 년도)	감독	포스터
인시디어스 (2010)	제임스 완	
한줄평	어릴 때는 무서웠는데 지금 다시 보니까 시시하다.	
평점	2	
명장면		
<div>수정</div> <div>삭제</div>		

[그림] 리뷰 수정 최종 상태(리뷰 보기로 이동) 화면

- 소스 코드

ReviewView.js의 일부 <pre>// 20200370 김혜진 const handleClickUpdate = (e) => { // 리뷰 수정 버튼 클릭이벤트 핸들러 window.location.href="/reviews/update/"+e.target.name; } ... 중략 ... <button class="btn btn-outline-primary" name={review.id} onClick={handleClickUpdate}>수정 </button></pre>
InputReview.js <pre>// 20200370 김혜진 import { useParams } from 'react-router-dom'; const { useState, useEffect } = require('react');</pre>

```

const axios = require('axios');

function UpdateReview() {
  const [reviewText, setReviewText] = useState('');
  const [grade, setGrade] = useState('');
  const [scene, setScene] = useState('');

  /* 입력 내용 핸들러 */
  const handleReviewText = (e) => { setReviewText(e.target.value) }
  const handleGrade = (e) => { setGrade(e.target.value) }
  const handleScene = (e) => { setScene(e.target.value) }

  useEffect(() => {
    getReview()
  }, [])

  let reviewId = useParams().reviewId; // 현재 경로의 reviewId 얻어오기 (App.js에서 path로 지정함)

  const getReview = async() => { // 이전 리뷰 내용 받아오기
    try {
      const res = await axios.get('http://localhost:3030/reviews/'+reviewId); // /reviews/:id 로 get요청 하여 기존 리뷰 내용 받아오기
      setReviewText(res.data[0].review_text);
      setGrade(res.data[0].grade);
      setScene(res.data[0].scene);
    } catch (e) {
      console.log(e);
    }
  };

  const handleClickUpdate = () => { // 수정 버튼 클릭이벤트 핸들러
    let password = prompt('비밀번호를 입력하세요');
    let obj = {
      review_text: reviewText,
      grade: grade,
      scene: scene
    };
    let config = {
      headers: { Authorization: password }
    };
    axios.put('http://localhost:3030/reviews/'+reviewId, obj, config) // 수정 내용, 비밀번호를 담아 /reviews/:id에 put 요청
    .then(res => {
      alert("성공!");
      window.location.href="/reviews/"+reviewId; // 성공 시 리뷰 보기로 이동
    })
    .catch(err => {
      alert("실패!");
    })
  }
}

```



```

return (
  <div class="container my-5">
    <div class="row">
      <div class="col h4">
        리뷰 수정
      </div>
    </div>
    <div class="row">
      <label for="reviewText" class="form-label col-3">한줄 평</label>
      <input
        class="form-control col"
        id="reviewText"
        value = {reviewText}
        onChange = {handleReviewText}
      />
    </div>
    <div class="row">
      <label for="movie_openYear" class="form-label col-3">평점</label>
      <input
        class="form-control col"
        id="grade"
        value = {grade}
        onChange = {handleGrade}
      />
    </div>
    <div class="row">
      <label for="scene" class="form-label col-3">명장면</label>
      <input
        class="form-control col"
        id="scene"
        value = {scene}
        onChange = {handleScene}
      />
    </div>

    <div class="row">
      <div class="col text-end">
        <button class="btn btn-outline-primary" onClick={handleClickUpdate}>수정
      </div>
    </div>
  </div>
);
}

export default UpdateReview;

```

review.js의 일부

```

// 20200370 김혜진
router.get('/:id', async (req, res) => {
  let id = req.params.id;
  let review = await (SendQuery("SELECT * from movie,review where movie.id =
review.movie_id and review.id = ?;", id));

```

```

    if (review !== null) {
      res.status(200).send(review);
    } else {
      res.status(400).end();
    }
  })
  ... (중략) ...
  router.put('/:id', async (req, res) => {
    if (adminPW !== req.headers.authorization) { // 비밀번호 확인 (관리자만)
      res.status(400).send();
      return;
    }

    let obj = [
      req.body.review_text,
      req.body.grade,
      req.body.scene,
      req.params.id
    ];

    let result = await SendQuery("UPDATE review SET review_text=?, grade=?, scene=? where id = ?", obj); // 3가지 내용만 UPDATE 요청
    if (result !== null) {
      console.log(result);
      res.status(200).send();
    }
    else {
      res.status(400).send();
    }
  })
}

```

- ReviewView.js에서 수정 버튼을 누르면 /reviews/update/:id로 이동한다.
- 기존 리뷰 내용을 get 요청 하여 받아와 미리 화면에 띄워둔다.
- 등록과 마찬가지로 관리자가 입력한 폼 내용과 비밀번호를 입력받아 /reviews/:id에 put으로 수정을 요청한다.
- 서버에서는 비밀번호를 확인하고 id로 조건을 주어 UPDATE 요청한다. 성공 시 200 코드를 반환한다.
- 비밀번호가 다르거나 UPDATE에 실패했다면 400 코드를 반환한다.
- 200 코드를 받으면 성공 알림 창을 띄우고 리뷰 보기 화면으로 이동한다.
- 400 코드를 받으면 실패 알림 창을 띄운다.

4.1.5 관리자 리뷰 삭제

- 실행 화면

localhost:3000/reviews/9

20200370 김혜진

localhost:3000 내용:
비밀번호를 입력하세요

제목
인사
(2010)



확인 취소

한줄평
어릴 때는 무서웠는데 지금 다시 보니까 시시하다.

평점
2

명장면

수정 삭제



[그림] 리뷰 삭제 초기 상태 화면과 중간 상태(비밀번호 입력) 화면


```
let result = await SendQuery("DELETE FROM review where id=?", req.params.id);
if (result != null) {
  console.log(result);
  res.status(200).send();
}
else {
  res.status(400).send();
}
})
```

- ReviewView.js에서 삭제 버튼을 누르면 handleClickDelete 함수가 호출된다.
- 함수에서는 id를 추출하고 비밀번호를 입력받아 /reviews/:id에 delete 요청을 보낸다.
- 서버에서는 파라미터의 id를 조건으로 주어 DELETE문 쿼리를 보낸다. 성공 시 200 코드를 반환한다.
- 비밀번호가 일치하지 않거나 DELETE 쿼리가 실패하면 400 코드를 반환한다.
- 200 코드를 받으면 성공 알림 창을 띄우고 리뷰 목록으로 이동하고, 400 코드를 받으면 실패 알림 창을 띄운다.

4.1.6 리뷰 리스트 보기

- 실행 화면



HORROR MOVIE REVIEW BLOG

[리뷰 목록 보기](#)

[리뷰 안 된 영화 목록 보기](#)

[전체 영화 목록 보기](#)

[영화 등록하기](#)

리뷰 목록

순번	포스터	제목(개봉년도)	감독	명장면
1		인사이드맨(2010)	제임스완	
한줄평: 어릴 때는 무서웠는데 지금 다시 보니까 시시하다. (2)				

ReviewList.js

// 20200370 김혜진

import { useState, useEffect } from 'react'

import './style.css';

import axios from 'axios'

const Review = (prop) => {

const handleClickView = (e) => { // /reviews/:id로 이동하여 리뷰 보기
window.location.href="/reviews/"+e.target.getAttribute('name');
}

const handleMouseOver = (e) => { // 마우스가 올라가면 회색으로 배경색 변경
e.target.style.background = '#eeeeee';
}

const handleMouseOut = (e) => { // 마우스가 나가면 배경색 없앰
e.target.style.background = 'none';
}

return (

<>

<tr>

<td rowSpan='2'>{prop.index+1}</td>

<td></td>

<td>{prop.data.title}({prop.data.open_year})</td>

<td>{prop.data.director}</td>

<td></td>

</tr>

<tr>

<td colspan='4' name={prop.data.id} onMouseOver={handleMouseOver}
onMouseOut={handleMouseOut} onClick={handleClickView}>

한줄평: {prop.data.review_text} ({prop.data.grade})

</td>

</tr>

</>

);

}

function ReviewList() {

let [reviews, setReviews] = useState([])

useEffect(() => {

getReviews()

}, [])

const getReviews = async () => { // 리뷰 받아오는 함수

try {

const res = await axios.get('http://localhost:3030/reviews/')

setReviews(res.data);

} catch (e) {

console.log(e)

}

}

```

return (
  <div class="container my-5">
    <div class="row">
      <div class="col h4">
        리뷰 목록
      </div>
    </div>
    <div class="row">
      <div class="col">
        <table class="table table-bordered">
          <thead>
            <tr>
              <th>순번</th>
              <th>포스터</th>
              <th>제목(개봉년도)</th>
              <th>감독</th>
              <th>명장면</th>
            </tr>
          </thead>
          <tbody>
            {reviews.map( (r, i) => <Review key={i} data={r} index={i}/>)}
          </tbody>
        </table>
      </div>
    </div>
  </div>
);
}

```

export default ReviewList

review.js의 일부

```

// 20200370 김혜진
router.get('/', async (req, res) => {
  // DB에서 리뷰 목록 가져 오기
  reviews = await SendQuery("SELECT * from movie, review where movie.id =
review.movie_id order by review.grade desc;", null);

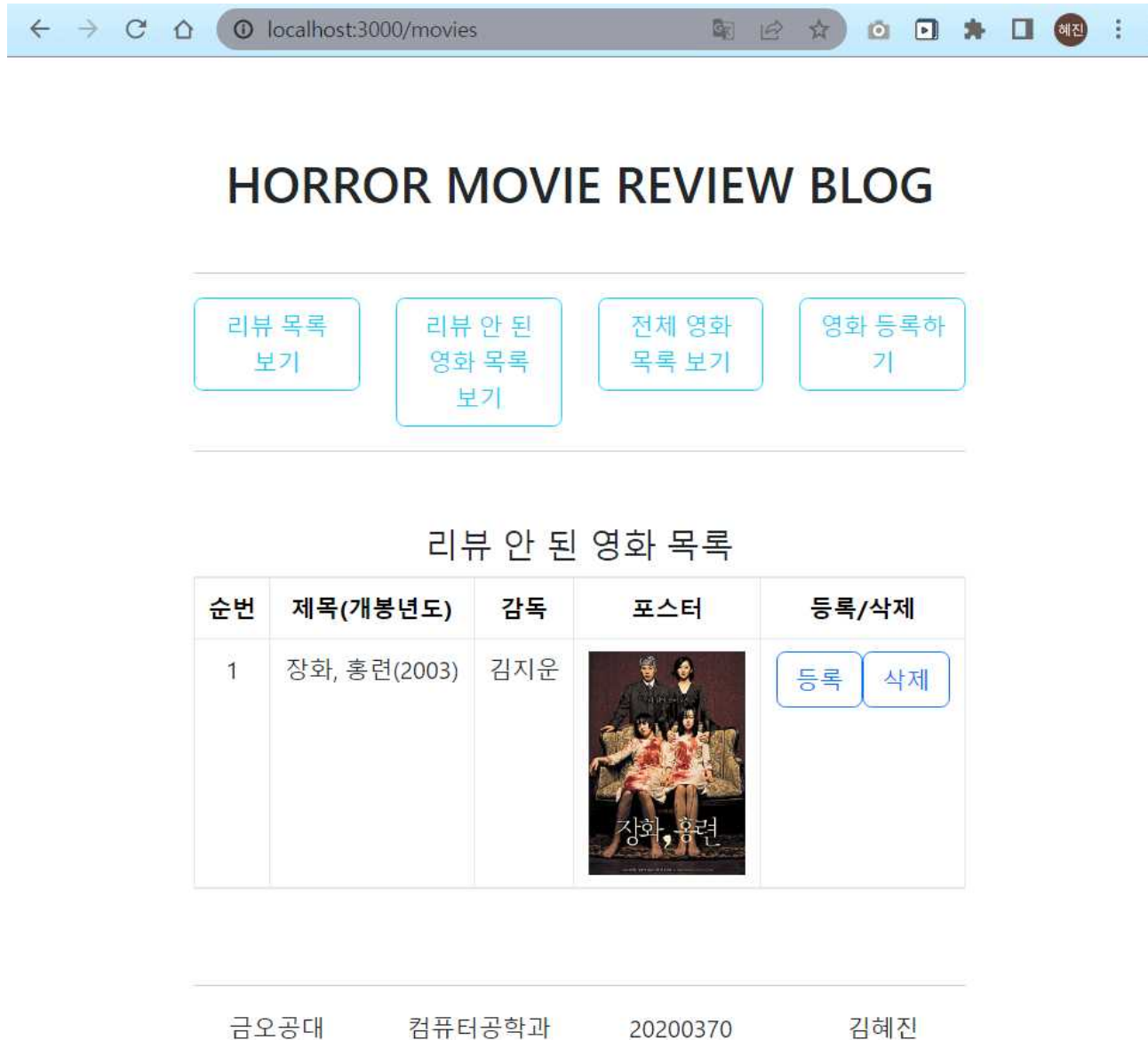
  if (reviews != null) {
    res.status(200).send(reviews);
  } else {
    res.status(400).end();
  }
})

```

- 리뷰를 얻기 위해 /reviews에 get 요청을 보냈다. 서버에서는 review 테이블에서 평점 내림차순으로 SELECT하여 응답을 보낸다.
- 리뷰 틀은 <Review /> 로 따로 만들었다. 여기서 리뷰 하나하나를 받아 테이블 형태로 띄운다.
- 한줄 평이 있는 셀을 클릭할 때 리뷰 보기로 이동해야 하므로 셀에 마우스가 올라가있을 때 색깔을 바꾸는 함수도 추가했다.

4.1.7 리뷰 안 된 영화 리스트 보기

- 실행 화면



[그림] 리뷰 안 된 영화 리스트 보기 화면

- 소스 코드

```
MovieList.js
// 20200370 김혜진
import { useState, useEffect } from 'react'
import axios from 'axios'

const Movie = (prop) => {

  const handleClickDelete = (e) => { // 삭제 버튼 클릭 이벤트 핸들러
    let password = prompt('비밀번호를 입력하세요');
    let config = {
      headers: { Authorization: password }
    };
  };
}
```



```

    axios.delete('http://localhost:3030/movies/'+e.target.name, config) // /movie/:id로 비밀번호
    호를 담아 delete 요청
    .then(res => {
      console.log(res);
      alert("성공!");
      window.location.href="/movies"; // 성공 시 /movies로 이동하여 새로그침
    })
    .catch(err => {
      alert("실패!");
      console.log(err);
    })
  }

  const handleClickInput = (e) => { // 등록 버튼 클릭 이벤트 핸들러
    window.location.href="/reviews/input/"+e.target.name; // /reviews/input/영화아이디 로 이
    동
  }

  return (
    <>
    <tr>
      <td>{prop.index+1}</td>
      <td>{prop.data.title}({prop.data.open_year})</td>
      <td>{prop.data.director}</td>
      <td><img src={prop.data.poster} style={{ height: '150px'}}></img></td>
      <td>
        <button class="btn btn-outline-primary" name={prop.data.id}
        onClick={handleClickInput}>등록</button>
        <button class="btn btn-outline-primary" name={prop.data.id}
        onClick={handleClickDelete}>삭제</button>
      </td>
    </tr>
    </>
  );
}

function MovieList() {
  let [movies, setMovies] = useState([])
  useEffect(() => {
    getMovies()
  }, [])

  const getMovies = async() => { // 영화 목록 받아오기
    try {
      let res = await axios.get('http://localhost:3030/movies?isNotReviewed=true')
      setMovies(res.data);
    } catch (e) {
      console.log(e)
    }
  };

  return (

```

```

<div class="container my-5">
  <div class="row">
    <div class="col h4">
      리뷰 안 된 영화 목록
    </div>
  </div>
  <div class="row">
    <div class="col">
      <table class="table table-bordered">
        <thead class="table-light">
          <tr>
            <th>순번</th>
            <th>제목(개봉년도)</th>
            <th>감독</th>
            <th>포스터</th>
            <th>등록/삭제</th>
          </tr>
        </thead>
        <tbody>
          {movies.map( (r, i) => <Movie key={i} data={r} index={i}/>)}
        </tbody>
      </table>
    </div>
  </div>
</div>
);
}

```

export default MovieList;

movie.js의 일부

// 20200370 김혜진

```

router.get('/', async (req, res) => {
  // DB에서 영화 목록 가져 오기
  let searchTitle = req.query.title;
  let movies;
  if (searchTitle) { // 검색할 타이틀이 있는 경우
    movies = await SendQuery("SELECT * from movie where title = ?", searchTitle); // title
    로 조건 주어 받아오기
  } else if (req.query.isNotReviewed == 'true') { // 리뷰 안 된 영화 목록을 요청하는 경우
    movies = await SendQuery("SELECT * from movie where id not in (select movie_id
    from review) order by input_date desc;", null); // 입력 날짜 순으로 정렬
  } else { // 전체 영화 목록을 요청하는 경우
    movies = await SendQuery("SELECT * from movie order by input_date;", null); // 입력
    날짜 순으로 정렬
  }

  if (movies != null) {
    res.status(200).send(movies);
  } else {
    res.status(400).end();
  }
})

```

- 리뷰 안 된 영화 목록을 보기 위해 /movies에 GET으로 요청하되, 파라미터인 isNotReviewed를 True로 보낸다.
- 그러면 서버에서는 isNotReviewed에는 True가 들어간다. 따라서 else if문이 실행된다.
- review에 있지 않은 movie들을 입력 날짜 내림차순(즉 최신순)으로 정렬하여 보낸다.
- 서버로부터 movie 리스트를 받으면 <Movie />태그로 데이터에 대한 테이블을 구성한다.
- 테이블의 마지막 열에는 영화를 등록/삭제할 수 있도록 버튼이 들어간다. 이는 각 핸들러에서 처리하며 등록 버튼을 누르면 해당 영화를 id로 하여 등록 창으로 이동한다.

4.1.8 전체 영화 리스트 보기

- 실행 화면

전체 영화 목록

순번	제목(개봉년도)	감독	포스터	등록
1	인시디어스(2010)	제임스 완		등록
2	장화, 홍련(2003)	김지운		등록

제목 검색

금오공대 컴퓨터공학과 20200370 김혜진
[그림] 전체 영화 리스트 보기 화면

MovieList.js

// 20200370 김혜진

import { useState, useEffect } from 'react'

import axios from 'axios'

const Movie = (prop) => {

const handleClickInput = (e) => { // 등록 버튼 클릭이벤트 핸들러
window.location.href="/reviews/input/"+e.target.name;
}

return (

<>

<tr>

<td>{prop.index+1}</td>

<td>{prop.data.title}({prop.data.open_year})</td>

<td>{prop.data.director}</td>

<td></td>

<td>

<button class="btn btn-outline-primary" name={prop.data.id}
onClick={handleClickInput}>등록</button>

</td>

</tr>

</>

);

}

function AllMovieList() {

let [movies, setMovies] = useState([])

let [title, setTitle] = useState('');

const handleTitle = (e) => { setTitle(e.target.value) }

useEffect(() => {

getMovies()

}, [])

const getMovies = async () => { // 리뷰 목록 가져오기

try {

let res = await axios.get('http://localhost:3030/movies?isNotReviewed=false') // /movies
에 isNotReviewed를 False로 해서 전체 영화 목록 가져오기

setMovies(res.data);

} catch (e) {

console.log(e)

}

};

const handleClickSearch = async () => { // 검색 버튼 클릭이벤트 핸들러

try {

let res = await axios.get('http://localhost:3030/movies?title='+title) // /movies에 title을 파
라미터로 주어 타이틀로 검색하기

setMovies(res.data);

} catch (e) {

```

    console.log(e)
  }
}

return (
  <div class="container my-5">
    <div class="row">
      <div class="col h4">
        전체 영화 목록
      </div>
    </div>
    <div class="row">
      <div class="col">
        <table class="table table-bordered">
          <thead class="table-light">
            <tr>
              <th>순번</th>
              <th>제목(개봉년도)</th>
              <th>감독</th>
              <th>포스터</th>
              <th>등록</th>
            </tr>
          </thead>
          <tbody>
            {movies.map( (r, i) => <Movie key={i} data={r} index={i}/>)}
          </tbody>
        </table>
      </div>
      <div class="row">
        <label for="movie_title" class="form-label col-3">제목 검색</label>
        <input
          class="form-control col"
          id="movie_title"
          value = {title}
          onChange = {handleTitle}
        />
        <button class="btn btn-outline-primary col-2" onClick={handleClickSearch}>검색
      </button>
      </div>
    </div>
  </div>
);
}

```

export default AllMovieList;

movie.js의 일부

// 20200370 김혜진

```

router.get('/', async (req, res) => {
  // DB에서 영화 목록 가져 오기
  let searchTitle = req.query.title;
  let movies;
  if (searchTitle) { // 검색할 타이틀이 있는 경우

```

```

    movies = await SendQuery("SELECT * from movie where title = ?", searchTitle); // title
    로 조건 주어 받아오기
  } else if (req.query.isNotReviewed == 'true') { // 리뷰 안 된 영화 목록을 요청하는 경우
    movies = await SendQuery("SELECT * from movie where id not in (select movie_id
    from review) order by input_date desc;", null); // 입력 날짜 순으로 정렬
  } else { // 전체 영화 목록을 요청하는 경우
    movies = await SendQuery("SELECT * from movie order by input_date;", null); // 입력
    날짜 순으로 정렬
  }

  if (movies != null) {
    res.status(200).send(movies);
  } else {
    res.status(400).end();
  }
}
})

```

- 최초 렌더링 시 리뷰 목록을 /movies에 get요청으로 가져온다. 전체 영화 목록이므로 파라미터 isNotReviewed는 false로 보낸다.
- 영화 목록은 map 함수를 이용하여 <Movies />에서 각각 영화에 대한 정보를 띄운다.
- 테이블의 마지막 열의 등록 버튼을 누르면 해당 영화에 대한 리뷰를 작성할 수 있게 이동한다.
- 전체 영화 목록을 요청하는 경우 movie 테이블의 전체 내용을 입력 날짜 순으로 정렬하여 보낸다.

4.1.9 영화 검색

- 실행 화면

- 영화 검색 초기 상태(검색 전) 화면: [그림] 전체 영화 리스트 보기 화면과 동일함

HORROR MOVIE REVIEW BLOG

리뷰 목록
보기

리뷰 안 된
영화 목록
보기

전체 영화
목록 보기

영화 등록하
기

전체 영화 목록

순번	제목(개봉년도)	감독	포스터	등록
1	인시디어스(2010)	제임스 완		등록

제목 검색

검색

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 영화 검색 최종 상태 화면

- 소스 코드

```

MovieList.js의 일부
// 20200370 김혜진
... (중략) ...
const handleClickSearch = async () => { // 검색 버튼 클릭이벤트 핸들러
  try {
    let res = await axios.get('http://localhost:3030/movies?title='+title) // /movies에 title을 파
라미터로 주어 타이틀로 검색하기
    setMovies(res.data);
  } catch (e) {
    console.log(e)
  }
}
... (중략) ...
<div class="row">

```

```

<label for="movie_title" class="form-label col-3">제목 검색</label>
<input
  class="form-control col"
  id="movie_title"
  value = {title}
  onChange = {handleTitle}
/>
  <button class="btn btn-outline-primary col-2" onClick={handleClickSearch}>검색
</button>
</div>
... (후략)

```

movie.js의 일부

```

// 20200370 김혜진
router.get('/', async (req, res) => {
  // DB에서 영화 목록 가져 오기
  let searchTitle = req.query.title;
  let movies;
  if (searchTitle) { // 검색할 타이틀이 있는 경우
    movies = await SendQuery("SELECT * from movie where title = ?", searchTitle); // title
    로 조건 주어 받아오기
  }
  ... (중략) ...

  if (movies != null) {
    res.status(200).send(movies);
  } else {
    res.status(400).end();
  }
})

```

- 4.1.8의 [전체 영화 목록 보기]와 같은 파일을 사용한다.
- 검색 내용을 작성하고 검색 버튼을 클릭하면 handleClickSearch 함수가 실행되어 /movies에 title 파라미터를 주어 get요청을 한다.
- 서버에서는 title 파라미터가 있으면 where로 조건을 주어 SELECT 쿼리문을 보낸다.
- 검색 결과가 비었거나 해당하는 목록이 있다면 그것을 200번 코드와 함께 전송한다.
- 검색에서 실패하여 SendQuery 결과로 null이 반환되면 400번 코드를 전송한다.
- 200번 코드를 받았다면 결과를 다시 영화 목록으로 세팅하여 다시 테이블만 띄워질 수 있게 한다.

4.1.10 리뷰 보기

- 실행 화면



리뷰 보기

제목(개봉 년도)	감독	포스터
인시디어스 (2010)	제임스 완	
한줄평	어릴 때는 무서웠는데 지금 다시 보니까 시시하다.	
평점	2	
명장면		
<input type="button" value="수정"/> <input type="button" value="삭제"/>		

댓글 목록		
작성자	내용	수정/삭제
1	1	<input type="button" value="수정"/> <input type="button" value="삭제"/>
<input type="text" value="닉네임"/> <input type="text" value="비밀번호"/>	<input type="text" value="댓글 내용"/>	<input type="button" value="등록"/>
작성자로 댓글 검색	<input type="text"/>	<input type="button" value="검색"/>

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 보기 화면

```
ReviewView.js
// 20200370 김혜진
import { useParams } from 'react-router-dom';
import { useState, useEffect } from 'react'
import axios from 'axios'

const CommentInput = () => { // 댓글 등록 태그
  let [commentText, setCommentText] = useState('');
  let [nickname, setNickname] = useState('');
  let [password, setPassword] = useState('');
  const handleCommentText = (e) => { setCommentText(e.target.value) }
  const handleNickname = (e) => { setNickname(e.target.value) }
  const handlePassword = (e) => { setPassword(e.target.value) }

  let reviewId = useParams().reviewId; // 리뷰 ID

  const handleClickCommentInput = () => { // 댓글 등록 버튼 클릭이벤트 핸들러
    let obj = {
      comment_text: commentText,
      nickname: nickname,
      password: password
    };
    axios.post('http://localhost:3030/comments/'+reviewId, obj) // /comments/:id에 post로 댓글
    // 삽입 요청
    .then(res => {
      alert("성공!");
      window.location.href="/reviews/"+reviewId; // 성공 시 새로그침
    })
    .catch(err => {
      alert("실패!");
    })
  }

  return (
    <div class="row my-1">
      <div class="col-3">
        <input
          class="form-control"
          id="comment_text"
          value = {nickname}
          onChange = {handleNickname}
          placeholder = "닉네임"
        />
        <input
          class="form-control"
          id="comment_text"
          value = {password}
          onChange = {handlePassword}
          type = "password"
          placeholder = "비밀번호"
        />
      </div>
    </div>
  )
}
```

```

</div>
<div class="row col">
  <input
    class="form-control"
    id="comment_text"
    value = {commentText}
    onChange = {handleCommentText}
    placeholder = "댓글 내용"
  />
</div>
<button class="btn btn-outline-primary col-2 mx-3" onClick={handleClickCommentInput}>
등록</button>
</div>
)
}

```

```
const Comment = (prop) => { // 댓글 하나 태그
```

```
  let reviewId = useParams().reviewId;
```

```
  const handleClickUpdate = (e) => { // 댓글 수정 버튼 클릭이벤트 핸들러
```

```
    let commentText = prompt('수정할 내용을 입력하세요');
```

```
    let password = prompt('비밀번호를 입력하세요');
```

```
    let config = {
```

```
      headers: { Authorization: password }
```

```
  };
```

```
  let obj = {
```

```
    comment_text: commentText
```

```
  };
```

```
  axios.put('http://localhost:3030/comments/'+e.target.name, obj, config) // 댓글 내용과 비
  밀번호를 담아 /comments/:id에 put 요청
```

```
  .then(res => {
```

```
    console.log(res);
```

```
    alert("성공!");
```

```
    window.location.href="/reviews/"+reviewId; // 성공 시 새로고침
```

```
  })
```

```
  .catch(err => {
```

```
    alert("실패!");
```

```
    console.log(err);
```

```
  })
```

```
}
```

```
const handleClickDelete = (e) => { // 댓글 삭제 버튼 클릭이벤트 핸들러
```

```
  let password = prompt('비밀번호를 입력하세요');
```

```
  let config = {
```

```
    headers: { Authorization: password }
```

```
  };
```

```
  axios.delete('http://localhost:3030/comments/'+e.target.name, config) // /comments/:id에
  비밀번호를 담아 delete 요청
```

```
  .then(res => {
```

```
    console.log(res);
```

```
    alert("성공!");
```

```

window.location.href="/reviews/"+reviewId; // 성공 시 새로그침
})
.catch(err => {
  alert("실패!");
  console.log(err);
})
}

return (
  <tr>
    <td width="20%">{prop.data.nickname}</td>
    <td>{prop.data.comment_text}</td>
    <td width="30%">
      <button class="btn btn-outline-primary" name={prop.data.id}
onClick={handleClickUpdate}>수정</button>
      <button class="btn btn-outline-primary" name={prop.data.id}
onClick={handleClickDelete}>삭제</button>
    </td>
  </tr>
)
}

const Comments = () => {
  let [comments, setComments] = useState([])
  let [writer, setWriter] = useState("");
  const handleWriter = (e) => { setWriter(e.target.value) }

  useEffect(() => {
    getComments()
  }, [])

  let reviewId = useParams().reviewId;

  const getComments = async() => { // 댓글 목록 태그
    try {
      const res = await axios.get('http://localhost:3030/comments/'+reviewId); //
//comments/:id로 id번 리뷰의 댓글 목록 get 요청
      setComments(res.data); // 성공 시 데이터를 set하여 띄울 수 있도록 함
    } catch (e) {
      console.log(e);
    }
  };

  const handleClickSearch = async () => { // 댓글 검색 버튼 클릭이벤트 핸들러
    try {
      let res = await
      axios.get('http://localhost:3030/comments/'+reviewId+'?nickname='+writer) // /comments/:id
로 닉네임 검색어 파라미터와 함께 get 요청
      setComments(res.data); // 성공 시 데이터를 set 하여 다시 띄울 수 있도록 함
    } catch (e) {
      console.log(e)
    }
  };
}

```

```

return (
  <>
    <tr>
      <th>작성자</th>
      <th>내용</th>
      <th>수정/삭제</th>
    </tr>
    {comments.map( (r, i) => <Comment key={i} data={r} index={i}/>)}
    <tr>
      <td colSpan='3'><CommentInput /></td>
    </tr>
    <tr>
      <td colSpan='3'>
        <div class='row mx-1'>
          <label for="comment_writer" class="form-label col-3">작성자로 댓글 검색</label>
          <input
            class="form-control col"
            id="comment_writer"
            value = {writer}
            onChange = {handleWriter}
          />
          <button class="btn btn-outline-primary col-2" onClick={handleClickSearch}>검색
        </button>
        </div>
      </td>
    </tr>
  </>
)
}

function ReviewView() { // 전체 리뷰 뷰
  let [review, setReview] = useState({})
  useEffect(() => {
    getReview()
  }, {})

  let reviewId = useParams().reviewId;

  const getReview = async() => {
    try {
      const res = await axios.get('http://localhost:3030/reviews/'+reviewId); // /reviews/:id로
리뷰 내용 get 요청
      setReview(res.data[0]);
    } catch (e) {
      console.log(e);
    }
  };

  const handleClickUpdate = (e) => { // 리뷰 수정 버튼 클릭이벤트 핸들러
    window.location.href="/reviews/update/"+e.target.name; // update 화면으로 이동
  }
}

```

```

const handleClickDelete = (e) => { // 리뷰 삭제 버튼 클릭이벤트 핸들러
let password = prompt('비밀번호를 입력하세요');
let config = {
  headers: { Authorization: password }
};
axios.delete('http://localhost:3030/reviews/'+e.target.name, config) // /reviews/:id에 비밀
번호를 담아 delete 요청
.then(res => {
  alert("성공!");
  window.location.href="/reviews"; // 성공 시 영화 목록으로 새로고침
})
.catch(err => {
  alert("실패!");
  console.log(err);
})
}

return (
<div class="container my-5">
<div class="row">
<div class="col h4">
리뷰 보기
</div>
</div>
<div class="row">
<div class="col">
<table class="table table-bordered">
<thead>
<tr>
<th>제목(개봉년도)</th>
<th>감독</th>
<th>포스터</th>
</tr>
</thead>
<tbody>
<tr>
<td>{review.title}{review.open_year}</td>
<td>{review.director}</td>
<td><img src={review.poster} style={{ height: '200px'}}></img></td>
</tr>
<tr>
<th>한줄평</th>
<td colSpan='2'>{review.review_text}</td>
</tr>
<tr>
<th>평점</th>
<td colSpan='2'>{review.grade}</td>
</tr>
<tr>
<th>명장면</th>
<td colSpan='2'>

```

```

        <img src={review.scene} style={{ width: '80%'}}></img>
      </td>
    </tr>
  </tr>
  <td colSpan='3' class="text-end">
    <button class="btn btn-outline-primary" name={review.id}
onClick={handleClickUpdate}>수정</button>
    <button class="btn btn-outline-primary" name={review.id}
onClick={handleClickDelete}>삭제</button>
  </td>
</tr>
</tbody>
</table>
</div>
</div>
<div class="row">
  <div class="col">
    <table class="table table-bordered">
      <thead>
        <tr>
          <th class="py-2" colSpan='3'>댓글 목록</th>
        </tr>
      </thead>
      <tbody>
        <Comments />
      </tbody>
    </table>
  </div>
</div>
</div>
);
}

```

export default ReviewView;

review.js의 일부

// 20200370 김혜진

```

router.get('/:id', async (req, res) => {
  let id = req.params.id;
  let review = await (SendQuery("SELECT * from movie,review where movie.id =
review.movie_id and review.id = ?;", id));

  if (review != null) {
    res.status(200).send(review);
  } else {
    res.status(400).end();
  }
})

```

comment.js의 일부

// 20200370 김혜진

```

router.get('/:id', async (req, res) => {
  // DB에서 댓글 목록 가져 오기
  let searchNickname = req.query.nickname;
  let reviews;

```

```

if (searchNickname) { // 검색어가 있는 경우 where문 추가하여 쿼리
  reviews = await SendQuery("SELECT id, review_id, comment_text, input_date, nickname
from comment where review_id = ? and nickname = ?", [req.params.id, searchNickname]);
} else { // 검색어가 없는 경우 전체 댓글 목록 쿼리
  reviews = await SendQuery("SELECT id, review_id, comment_text, input_date, nickname
from comment where review_id = ? order by input_date;", req.params.id);
}

if (reviews != null) {
  res.status(200).send(reviews);
} else {
  res.status(400).end();
}
}
})

```

- ReviewView.js의 내용은 댓글 부분도 포함하고 있어 소스 코드가 길다.
 - ReviewView 태그 안에서 리뷰 내용을 출력하고, Comments 태그 안에서 댓글 목록을 관리한다.
- Comment 태그 안에서 각 댓글에 대한 테이블 행 출력 및 수정/삭제 기능을 담당한다. CommentInput 태그 안에서 댓글 등록에 대한 관리를 한다.
- 최초 페이지 렌더링 시 get 요청을 두 번 보내어 리뷰 내용과 댓글 목록을 가져온다. 한꺼번에 가져올 수도 있었는데 join 시 리뷰 내용도 여러 번 나올 수 있기에 따로 요청하는 것으로 구현했다.

4.1.11 리뷰 댓글 등록

- 실행 화면

The screenshot shows a web browser window with the address bar displaying 'localhost:3000/reviews/10'. The page content is as follows:

한줄평	어릴 때는 무서웠는데 지금 다시 보니까 시시하다.
평점	2
명장면	
<div>수정</div> <div>삭제</div>	

Below the review, there is a section titled '댓글 목록' (Comment list). It contains a form for adding a comment:


작성자	내용	수정/삭제
<div>닉네임</div> <div>비밀번호</div>	<div>댓글 내용</div>	<div>등록</div>
<div>작성자로 댓글 검색</div>	<div></div>	<div>검색</div>

At the bottom of the page, there is a footer with the text: '금오공대 컴퓨터공학과 20200370 김혜진'.

[그림] 리뷰 댓글 등록 초기 상태 화면

댓글 목록		
작성자	내용	수정/삭제
<div>ESTJ</div> <div>....</div>	저는 이거 엄청 무섭게 봤는데 ㄷㄷ	등록
<div>작성자로 댓글 검색</div>		검색

localhost:3000/reviews/10

평점	2	
명장면		
		수정 삭제

댓글 목록

작성자	내용	수정/삭제
ESTJ	저는 이거 엄청 무섭게 봤는데 ㄷ ㄷ	수정 삭제

닉네임
비밀번호

댓글 내용

등록

작성자로 댓글 검색

검색

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 댓글 등록 최종 상태(새로 고침) 화면

- 소스 코드

```

ReviewView.js의 일부
// 20200370 김혜진
const CommentInput = () => { // 댓글 등록 태그
  let [commentText, setCommentText] = useState('');
  let [nickname, setNickname] = useState('');
  let [password, setPassword] = useState('');
  const handleCommentText = (e) => { setCommentText(e.target.value) }
  const handleNickname = (e) => { setNickname(e.target.value) }
  const handlePassword = (e) => { setPassword(e.target.value) }

  let reviewId = useParams().reviewId; // 리뷰 ID

  const handleClickCommentInput = () => { // 댓글 등록 버튼 클릭이벤트 핸들러

```

```

let obj = {
  comment_text: commentText,
  nickname: nickname,
  password: password
};
axios.post('http://localhost:3030/comments/'+reviewId, obj) // /comments/:id에 post로 댓글
글 삽입 요청
.then(res => {
  alert("성공!");
  window.location.href="/reviews/"+reviewId; // 성공 시 새로그침
})
.catch(err => {
  alert("실패!");
})
}

return (
<div class="row my-1">
<div class="col-3">
<input
  class="form-control"
  id="comment_text"
  value = {nickname}
  onChange = {handleNickname}
  placeholder = "닉네임"
/>
<input
  class="form-control"
  id="comment_text"
  value = {password}
  onChange = {handlePassword}
  type = "password"
  placeholder = "비밀번호"
/>
</div>
<div class="row col">
<input
  class="form-control"
  id="comment_text"
  value = {commentText}
  onChange = {handleCommentText}
  placeholder = "댓글 내용"
/>
</div>
<button class="btn btn-outline-primary col-2 mx-3" onClick={handleClickCommentInput}>
등록</button>
</div>
)
}

```

comment.js의 일부

// 20200370 김혜진

router.post('/:id', async (req, res) => { // /comments/:id에 post로 오는 요청

```
let commentObj = {
  review_id: req.params.id,
  comment_text: req.body.comment_text,
  nickname: req.body.nickname,
  password: req.body.password,
  input_date: new Date()
};

if (await SendQuery("INSERT INTO comment SET ?;", commentObj)) // 삽입 query문
  res.status(200).send();
else
  res.status(400).send();
})
```

• 닉네임, 비밀번호, 내용을 입력하고 등록버튼을 누르면 등록 버튼 클릭이벤트 핸들러에서 데이터를 모아 post요청으로 보낸다.

• 서버에서 post 요청을 받으면 데이터를 comment 테이블 명세에 맞게 객체를 만들고 insert 쿼리문을 보낸다.

• 성공 시 200 코드를 반환하고, 실패 시 400 코드를 반환한다.

• 200 코드를 받으면 성공 알림 창을 띄우고 새로그침하며 실패 시 실패 알림 창을 띄운다.

4.1.12 리뷰 댓글 수정

- 실행 화면

- 리뷰 댓글 수정 초기 상태: [그림]와 동일

localhost:3000 내용:
수정할 내용을 입력하세요

저는 이거 엄청 무섭게 봤는데 덜덜..

확인 취소

수정 삭제

댓글 목록		
작성자	내용	수정/삭제
ESTJ	저는 이거 엄청 무섭게 봤는데 ㄷ ㄷ	수정 삭제

닉네임
비밀번호

댓글 내용

등록

작성자로 댓글 검색

검색

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 댓글 수정 중간 상태(수정할 내용 입력) 화면

localhost:3000/reviews/10

localhost:3000 내용:

비밀번호를 입력하세요

확인

취소

수정

삭제

댓글 목록

작성자	내용	수정/삭제
ESTJ	저는 이거 엄청 무섭게 봤는데 ㄷ ㄷ	<div>수정</div> <div>삭제</div>

닉네임

비밀번호

댓글 내용

등록

작성자로 댓글 검색

검색

금오공대 컴퓨터공학과 20200370 김혜진
 [그림] 리뷰 댓글 수정 중간 상태(비밀번호 입력) 화면


```

};
axios.put('http://localhost:3030/comments/'+e.target.name, obj, config) // 댓글 내용과 비
밀번호를 담아 /comments/:id에 put 요청
.then(res => {
  console.log(res);
  alert("성공!");
  window.location.href="/reviews/"+reviewId; // 성공 시 새로고침
})
.catch(err => {
  alert("실패!");
  console.log(err);
})
}

```

comment.js의 일부

// 20200370 김혜진

```

router.put('/:id', async (req, res) => { // /comments/:id에 put 요청 처리
  let password = await SendQuery("SELECT password FROM comment where id=?",
  req.params.id) // DB 비밀번호 받아오기
  if (password[0].password !== req.headers.authorization) { // 비밀번호 확인 (등록한 사람만)
    res.status(400).send();
    return;
  }
  let result = await SendQuery("UPDATE comment SET comment_text=? where id=?",
  [req.body.comment_text, req.params.id]); // update 쿼리문
  if (result !== null) {
    console.log(result);
    res.status(200).send();
  }
  else {
    res.status(400).send();
  }
})

```

- 댓글 수정 버튼을 누르면 알림 창으로 댓글 수정 내용과 댓글 작성 시 등록한 비밀번호를 입력한다.
- 데이터를 담아 /comments/:id에 put으로 요청한다.
- 요청을 받은 서버는 패스워드 확인을 위해 DB로부터 댓글의 비밀번호를 가져와 비교한다.
- 비밀번호가 일치하면 UPDATE 쿼리문을 보낸다.
- UPDATE에 성공하면 200 코드를 반환하고, 실패하거나 비밀번호가 다르면 400 코드를 반환한다.
- 200 코드를 받으면 성공 알림 창을 띄우고 새로 고침하며, 400 코드를 받으면 실패 알림 창을 띄운다.

4.1.13 리뷰 댓글 삭제

- 실행 화면

- 리뷰 댓글 삭제 초기 상태 화면: [그림]과 같다.

localhost:3000/reviews/10

localhost:3000 내용:

비밀번호를 입력하세요

확인

취소

수정

삭제

댓글 목록

작성자	내용	수정/삭제
ESTJ	저는 이거 엄청 무섭게 봤는데 덜 덜...	<div>수정</div> <div>삭제</div>

닉네임

비밀번호

댓글 내용

등록


작성자로 댓글
검색

검색

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 댓글 삭제 중간 상태(비밀번호 입력) 화면

localhost:3000/reviews/10

한줄평	어릴 때는 무서웠는데 지금 다시 보니까 시시하다.	
평점	2	
명장면		
		<div>수정</div> <div>삭제</div>

댓글 목록

작성자	내용	수정/삭제
<div>닉네임</div> <div>비밀번호</div>	<div>댓글 내용</div>	<div>등록</div>
<div>작성자로 댓글 검색</div>	<div></div>	<div>검색</div>

금오공대 컴퓨터공학과 20200370 김혜진

[그림] 리뷰 댓글 삭제 최종 상태 화면

- 소스 코드

```

ReviewView.js의 일부
// 20200370 김혜진
const handleClickDelete = (e) => { // 댓글 삭제 버튼 클릭이벤트 핸들러
  let password = prompt('비밀번호를 입력하세요');
  let config = {
    headers: { Authorization: password }
  };
  axios.delete('http://localhost:3030/comments/'+e.target.name, config) // /comments/:id에
  비밀번호를 담아 delete 요청
  .then(res => {
    console.log(res);
    alert("성공!");
    window.location.href="/reviews/"+reviewId; // 성공 시 새로고침
  });
}

```

```

})
.catch(err => {
  alert("실패!");
  console.log(err);
})
}

```

comment.js의 일부

```

// 20200370 김혜진
router.delete('/:id', async (req, res) => {
  let password = await SendQuery("SELECT password FROM comment where id=?",
  req.params.id) // 비밀번호 가져오기
  if (password[0].password !== req.headers.authorization && req.headers.authorization !==
  adminPW) { // 비밀번호 확인(작성자, 관리자)
    res.status(400).send();
    return;
  }

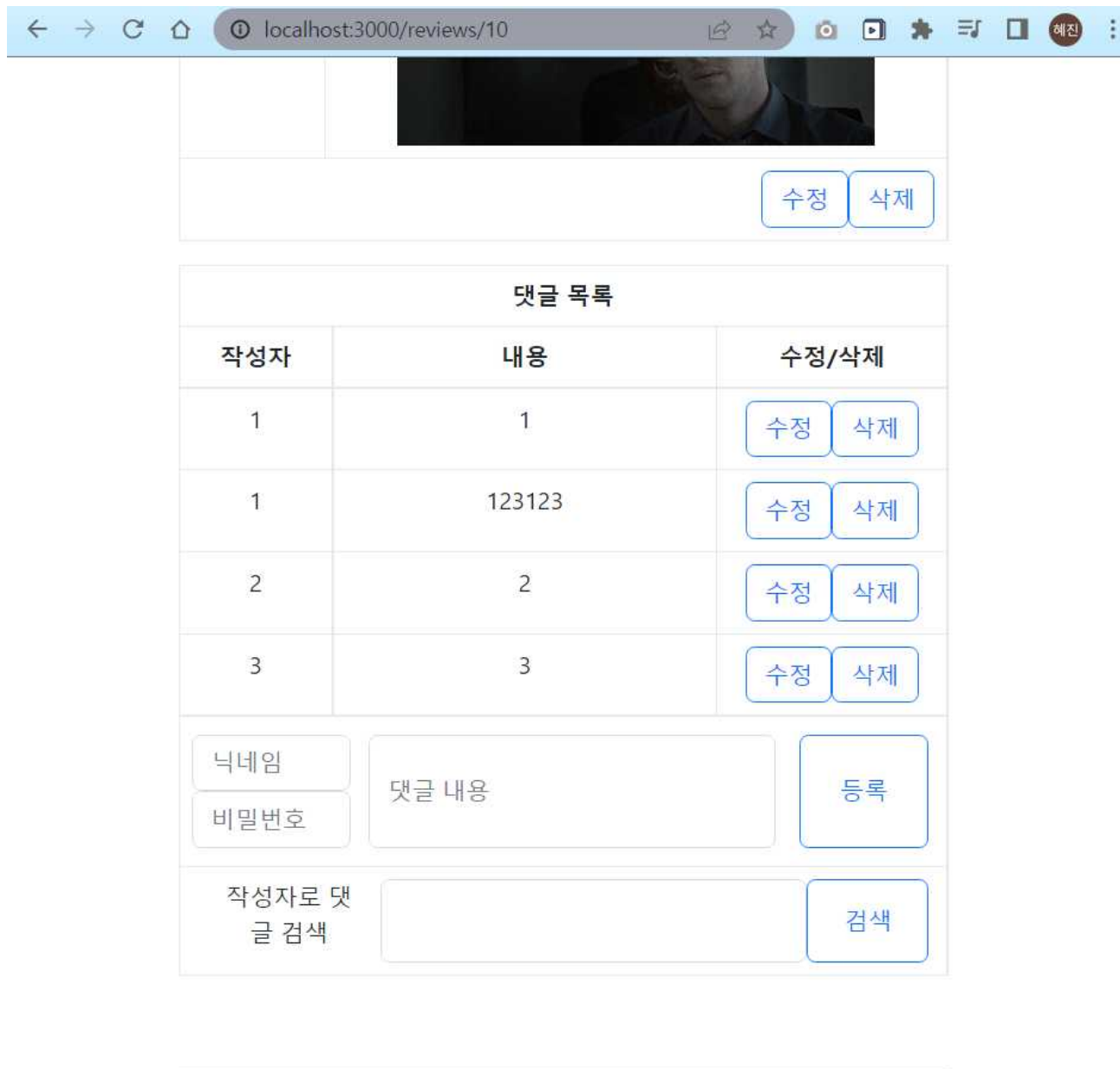
  let result = await SendQuery("DELETE FROM comment where id=?", req.params.id); // 삭
  제 쿼리문 보내기
  if (result !== null) {
    console.log(result);
    res.status(200).send();
  }
  else {
    res.status(400).send();
  }
})

```

- 삭제 버튼을 누르면 삭제 버튼 핸들러에서 비밀번호를 입력받고 /comments/:id에 delete 요청을 보낸다.
- 서버에서 요청을 받으면 DB에 있는 비밀번호를 받아오고, 비밀번호를 확인한다.
- 댓글 삭제는 작성자 뿐만 아니라 관리자도 가능하게 했다.
- 비밀번호가 일치하면 파라미터의 id를 이용하여 DELETE 쿼리문을 보낸다.
- DELETE가 성공하면 200 코드를 반환하고, 실패하거나 비밀번호가 다르면 400 코드를 반환한다.
- 200 코드를 반환받으면 성공 알림 창을 띄우고 새로 고침 하며, 400 코드를 반환받으면 실패 알림 창을 띄운다.

4.1.14 리뷰 댓글 검색

- 실행 화면



localhost:3000/reviews/10

수정 삭제

댓글 목록		
작성자	내용	수정/삭제
1	1	수정 삭제
1	123123	수정 삭제
2	2	수정 삭제
3	3	수정 삭제

닉네임
비밀번호

댓글 내용

등록

작성자로 댓글 검색

검색

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 댓글 검색 초기 상태 화면

localhost:3000/reviews/10

수정

삭제

댓글 목록

작성자	내용	수정/삭제
1	1	<div>수정</div> <div>삭제</div>
1	123123	<div>수정</div> <div>삭제</div>

닉네임

비밀번호

댓글 내용

등록

작성자로 댓글 검색

1

검색

금오공대

컴퓨터공학과

20200370

김혜진

[그림] 리뷰 댓글 검색 최종 상태 화면

- 소스 코드

<div> <div>ReviewView.js의 일부</div> <div> // 20200370 김혜진 const handleClickSearch = async () => { // 댓글 검색 버튼 클릭이벤트 핸들러 try { let res = axios.get('http://localhost:3030/comments/'+reviewId+'?nickname='+writer) // /comments/:id로 닉네임 검색어 파라미터와 함께 get 요청 setComments(res.data); // 성공 시 데이터를 set 하여 다시 띄울 수 있도록 함 } catch (e) { console.log(e) } } }; </div> </div>
<div> <div>comment.js의 일부</div> <div> // 20200370 김혜진 router.get('/:id', async (req, res) => { </div> </div>

```

// DB에서 댓글 목록 가져 오기
let searchNickname = req.query.nickname;
let reviews;
if (searchNickname) { // 검색어가 있는 경우 where문 추가하여 쿼리
  reviews = await SendQuery("SELECT id, review_id, comment_text, input_date, nickname
from comment where review_id = ? and nickname = ?", [req.params.id, searchNickname]);
} else { // 검색어가 없는 경우 전체 댓글 목록 쿼리
  reviews = await SendQuery("SELECT id, review_id, comment_text, input_date, nickname
from comment where review_id = ? order by input_date;", req.params.id);
}

if (reviews != null) {
  res.status(200).send(reviews);
} else {
  res.status(400).end();
}
})

```

- 댓글 검색어를 입력하고 검색 버튼을 누르면 검색 버튼 이벤트 핸들러가 수행된다.
- /comments/:id?nickname=검색어 형태로 get 요청을 보낸다.
- 서버는 요청을 받아 검색어를 조건절로 하여 select 문을 보낸다.
- 댓글 관련 select문은 비밀번호를 빼고 받아오기 위해 *를 쓰지 않고 애트리뷰트명을 입력해주어 받아왔다.
- 리뷰 내용이 없거나 있으면 리뷰 내용과 함께 200 코드를 반환하고, select 문에 오류가 있으면 400 코드를 반환한다.
- 200 코드를 받으면 검색한 내용을 띄운다.

5. 결 론

- 개인 기록용 블로그를 만드는 것이 목표였다. 실세계의 영화를 등록하여 리뷰를 작성하고 리뷰 별로 댓글을 작성할 수 있게 했다. 영화 등록/삭제/검색, 리뷰 등록/수정/삭제, 댓글 등록/수정/삭제/검색 기능을 구현하였다.
- 향후 개선을 한다면 실세계에 있는 영화만 등록할 수 있도록 API를 이용하여 검색 내용을 받아오는 것이 중요할 것 같다. 아무래도 관리자 비밀번호를 이용하면 보안 위험이 적긴 하지만 그래도 뚫릴 수 있고, 실세계에 없는 영화를 입력할 수도 있기 때문이다.
- 공포 영화 리뷰를 통해 내 리뷰를 기록할 수 있으며 한 영화에 여러 리뷰를 달수도 있으므로 여러 번 보았을 때에 달라진 생각과 평점을 확인할 수 있다. 다른 유저들이 내 리뷰나 영화에 대한 코멘트도 남길 수 있으므로 사람들과 의견도 공유할 수 있다. 따라서 영화를 보지 않은 사람에게는 영화를 볼 지를 선택할 때 있어서 도움이 될 수 있을 것이며 다양한 관점에서 볼 수 있다는 것이 좋을 것 같다.

6. 자체 분석과 평가

- 수행 과정에서는 별로 어렵지는 않았다. 웹 프론트엔드와 백엔드 경험이 있어서 쉽게 구현할 수 있었다. 하지만 조금 더 시간을 들였다면 설계서에 작성된 내용을 모두 구현할 수 있었을 텐데 아쉬웠다.
- 설계서에 작성했지만 시간 상 구현하지 못한 점은 다음과 같다.
 - 관리자 영화 등록 - 영화를 네이버 API로 검색할 수 있게 하여 실세계에 존재하는 영화만 등록하는 것으로 설계했지만 구현하지 못했다.
 - 알림 창 - 정말로 삭제하시겠습니까? 라는 문구를 넣게 설계했는데, 모든 삭제 및 성공 알림창에서 성공, 실패 등 한 단어로 대체하였다.
 - 리뷰 안 된 영화 리스트 보기 - 설계서에서는 페이지를 나누어 일정 개수만 출력하였고 리스트의 정렬 순서도 드롭박스로 변경할 수 있는 기능을 설계했지만 구현하지 못했다.
 - 목록 버튼 - 설계서에서는 리뷰 등록, 리뷰 수정, 리뷰 보기 등에서 목록 버튼이 존재했지만 모든 페이지에 헤더와 내비게이션 바가 존재하기 때문에 필요 없는 기능이라고 생각해서 뺐다.
 - 이미지 BLOB 타입 구현 - BLOB 타입으로 저장하면 불러와서 띄우는 것에 대해서 여러 자료를 찾아봤지만 js 파일로 구현하는 것이 직관적으로 바로 이해되지 않아서 그냥 이미지 주소를 저장하고 img 태그의 src로 띄우게 했다.
- 과제를 수행하면서 가장 어려운 것은 아무래도 디자인이었다. 디자인을 어떻게 해야 좋을 지를 많이 고민했는데, 이미 있는 블로그 디자인을 가져와 쓸 수도 있었지만 단순하게 구현하기 위해 부트스트랩의 컴포넌트와 레이아웃들을 이용하는 것으로 했다. 그리드를 이용하여 창 크기가 변경되어도 자연스럽게 볼 수 있도록 구현했다.
- 그리고 가능하면 태그를 많이 만들어 다른 기능에 독립적으로 구현할 수 있도록 하려고 했다. 하지만 실제로 태그가 많아지다 보니 너무 헛갈리는 점이 많았다. 특히 ReviewView.js에서 리뷰 내용과 댓글 목록/등록/검색 기능을 다 넣으려고 하다 보니 어려웠던 것 같다. 이 점은 파일을 따로 Comment.js로 구분했다면 더 쉽게 할 수 있었을 것 같다. 이 생각을 설계 당시 하지 않은 것은 아니지만 그렇게 되면 리뷰를 받아오고 리뷰 아이디를 받아오는 걸 ReviewView.js에서 하는데 Comment.js 파일로 어떻게 넘겨줄 지를 잘 모르겠어서 구현을 해보지는 못한 것 같다. 기회가 된다면 이런 디테일한 부분들도 공부해보고 싶다.

7. 부록

- 개발 일지
 - 6/5 구현 시작
 - 6/11 구현 완료
 - 6/16 최종 보고서 초본 작성
 - 6/21 최종 보고서 완성

- 기타 자료

- 서버 코드 src/dbconnect.js - 쿼리문을 보낼 때 이용한 SendQuery() 함수

```
const mysql = require('mysql');
const dbInfo = require('./secret.js').dbInfo;

const init = () => {
  return mysql.createConnection({
    host: dbInfo.host,
    user: dbInfo.user,
    password: dbInfo.password,
    database: dbInfo.database,
    port: dbInfo.port
  });
}

let dbConn = init();

exports.SendQuery = (sql, obj) => {
  return new Promise((resolve, reject) => {
    dbConn.query(sql, obj, (err, rows) => {
      if (!err) {
        console.log(sql, obj, "connect success");
        resolve(rows);
      }
      else {
        console.log(err);
        dbConn = init();
        resolve(null);
      }
    });
  });
}
```

- 서버 코드 src/api/index.js - 라우팅을 통해 movie.js 내용은 /movies 경로로, review.js 내용은 /reviews 경로로, comment.js 내용은 /comments 경로로 요청 받게끔 하였다.

```
const express = require('express');
const router = express.Router();

router.use('/movies', require('./movie.js'));

router.use('/reviews', require('./review.js'));

router.use('/comments', require('./comment.js'));

module.exports = router;
```

- 리액트 코드 - 라우트 경로를 지정해주었다.

```
import {BrowserRouter, Routes, Route} from "react-router-dom";
import Header from './Header.js';
import MovieList from './MovieList.js';
```



```

import InputMovie from './InputMovie.js';
import ReviewList from './ReviewList.js';
import InputReview from './InputReview.js';
import UpdateReview from './UpdateReview.js';
import ReviewView from './ReviewView.js';
import AllMovieList from './AllMovieList.js';
import Main from './Main.js';
import Footer from './Footer.js';

function App() {
  return (
    <BrowserRouter>
      <Header />
      <Routes>
        <Route path="/" element={<Main />} />
        <Route path="movies" element={<MovieList />} />
        <Route path="movies/input" element={<InputMovie />} />
        <Route path="reviews" element={<ReviewList />} />
        <Route path="reviews/:reviewId" element={<ReviewView />} />
        <Route path="reviews/input/:movieId" element={<InputReview />} />
        <Route path="reviews/update/:reviewId" element={<UpdateReview />} />
        <Route path="allMovies" element={<AllMovieList />} />
      </Routes>
      <Footer />
    </BrowserRouter>
  );
}
export default App;

```

참고문헌

- [1] Axios API, https://axios-http.com/docs/api_intro
- [2] Express.js 라우트 요청 객체(req), 응답 객체(res) 정리 <https://luckyowu.tistory.com/346>
- [3] React Router의 Link 컴포넌트 밑줄을 제거하는 방법은 무엇입니까?
<http://daplus.net/javascript-react-router%EC%9D%98-link-%EC%BB%B4%ED%8F%AC%EB%84%8C%ED%8A%B8-%EB%B0%91%EC%A4%84%EC%9D%84-%EC%A0%9C%EA%B1%B0%ED%95%98%EB%8A%94-%EB%B0%A9%EB%B2%95%EC%9D%80-%EB%AC%B4%EC%97%87%EC%9E%85/>
- [4] 자바스크립트(JavaScript) 입력 - prompt() 함수, confirm() 함수
<https://gangzzang.tistory.com/entry/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8JavaScript-%EC%9E%85%EB%A0%A5-prompt-%ED%95%A8%EC%88%98-confirm-%ED%95%A8%EC%88%98>
- [5] CSS 이미지 사이즈 맞추기
<https://gangzzang.tistory.com/entry/%EC%9E%90%EB%B0%94%EC%8A%A4%ED%81%AC%EB%A6%BD%ED%8A%B8JavaScript-%EC%9E%85%EB%A0%A5-prompt-%ED%95%A8%EC%88%98-confirm-%ED%95%A8%EC%88%98>