

# PJT 07

# DB 설계를 활용한 REST API 설계

# INDEX

---

- 개요
- 준비사항
- 요구사항
- 도전 과제
- 제출

# 개요

## 프로젝트 개요

- 영화 관련 데이터를 제공하는 RESTful API 서버 구성 단계
- 영화, 배우, 리뷰 데이터의 조회가 가능한 API 서버
- 리뷰 데이터 생성, 수정, 삭제가 가능한 API 서버
- 영화, 배우, 리뷰 간의 모델 관계가 형성된 애플리케이션

## 프로젝트 목표

- Django REST framework를 활용한 API 서버 제작
- HTTP request methods에 대한 이해
- HTTP response status codes에 대한 이해
- Many to one relationship(N:1)에 대한 이해
- Many to many relationship(N:M)에 대한 이해

# 준비사항

## | 개발도구

- Visual Studio Code
- Google Chrome
- Django 4.2.x
- Django REST framework
- Postman



# 초기 데이터 안내

## | 초기 데이터

1. 주어진 **fixture** 파일을 **movies** 앱 내에 위치시킴
2. fixture data의 load는 반드시 모델 정의 및 migrate 이후에 진행

```
$ python manage.py migrate
```

```
# json 파일은 movies/fixtures/movies/ 에 위치
```

```
$ python manage.py loaddata movies/actors.json movies/movies.json movies/reviews.json
```

# 요구사항

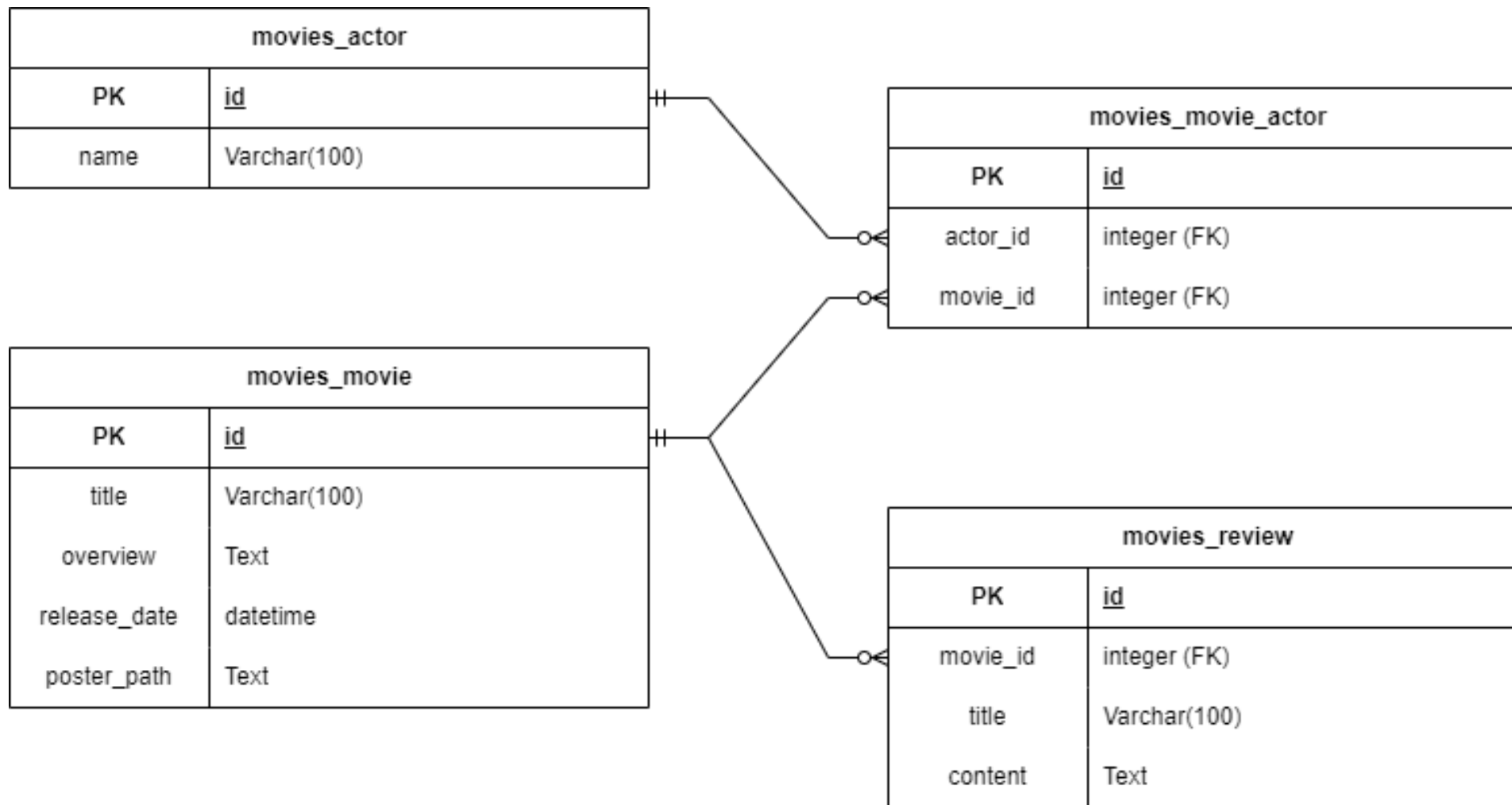
## | 필수 요구사항 - 프로젝트 및 앱

- 프로젝트 이름
  - mypjt
- 앱 이름
  - movies

## | 필수 요구사항 - Model Class

- Actor 모델 클래스
  - 배우 이름을 저장할 필드 1개 지정
- Movie 모델 클래스
  - 영화 제목, 줄거리, 개봉일, 포스터 주소를 저장할 필드 4개 지정
- Review 모델 클래스
  - 리뷰 제목, 내용을 입력할 필드 2개 지정

### 필수 요구사항 - ERD



# 모델 별 기능 구현

## | 필수 요구사항 - 모델 별 기능 구현

### A. Actor

- 배우 데이터 조회

### B. Movie

- 영화 데이터 조회

### C. Review

- 리뷰 데이터 조회 / 생성 / 수정 / 삭제

❖ 각 데이터는 JSON 데이터 타입을 따름



## 필수 요구사항 - view 함수

- movies 앱의 view 함수

함수명	역할	허용 HTTP Method
actor_list	• 전체 배우 목록 제공	GET
actor_detail	• 단일 배우 정보 제공 (출연 영화 제목 포함)	GET
movie_list	• 전체 영화 목록 제공	GET
movie_detail	• 단일 영화 정보 제공 (출연 배우 이름과 리뷰 목록 포함)	GET
review_list	• 전체 리뷰 목록 제공	GET
review_detail	• 단일 리뷰 조회 & 수정 & 삭제 (출연 영화 제목 포함)	GET / PUT / DELETE
create_review	• 리뷰 생성	POST

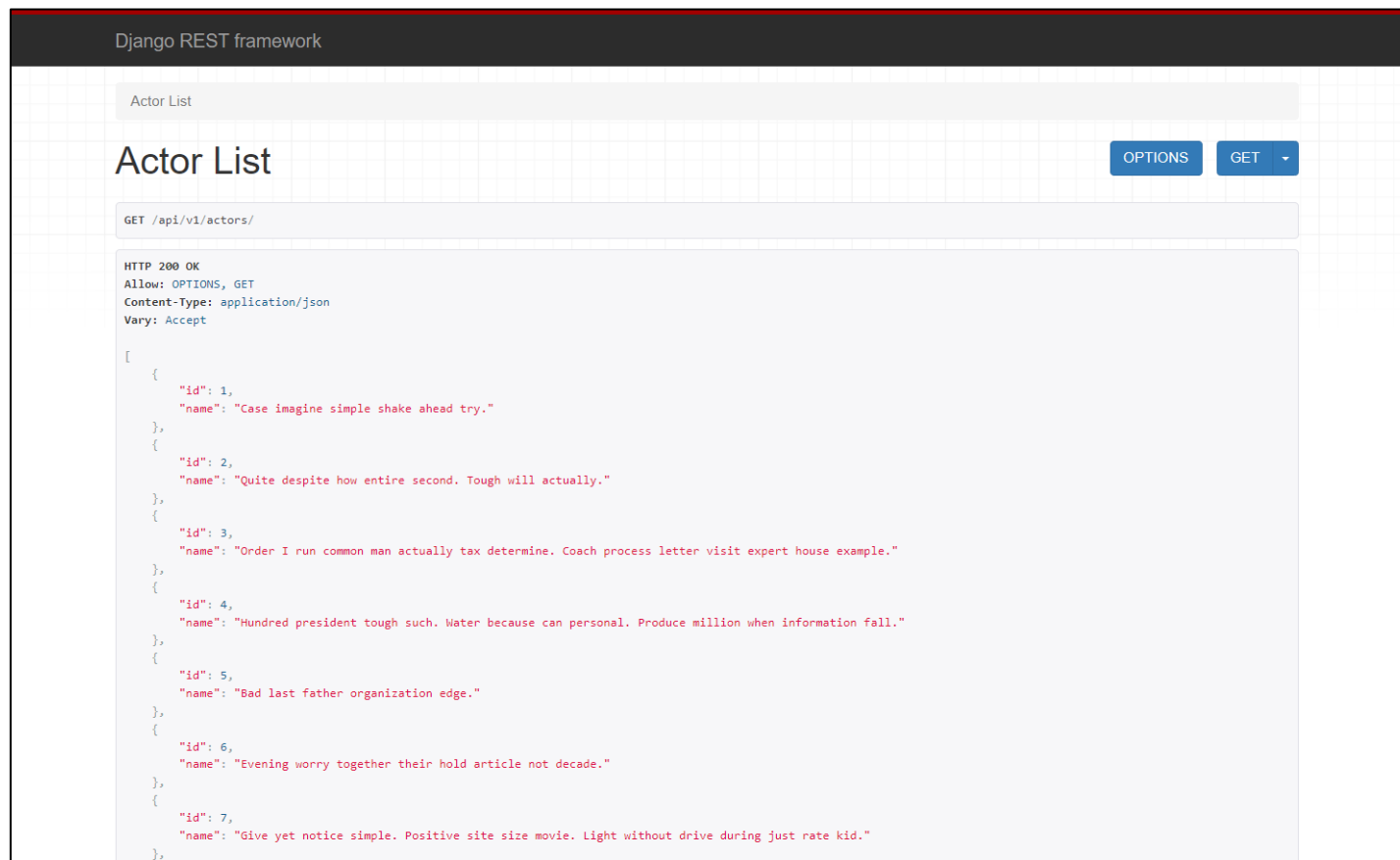
# 완성 응답 예시

## | 완성 응답 예시

- A. 전체 배우 목록 제공
- B. 단일 배우 정보 제공
- C. 전체 영화 목록 제공
- D. 단일 영화 정보 제공
- E. 전체 리뷰 목록 제공
- F. 단일 리뷰 조회 & 수정 & 삭제
- G. 리뷰 생성

## A. 전체 배우 목록 제공

- [GET] api/v1/actors/



## B. 단일 배우 정보 제공

- [GET] api/v1/actors/1/

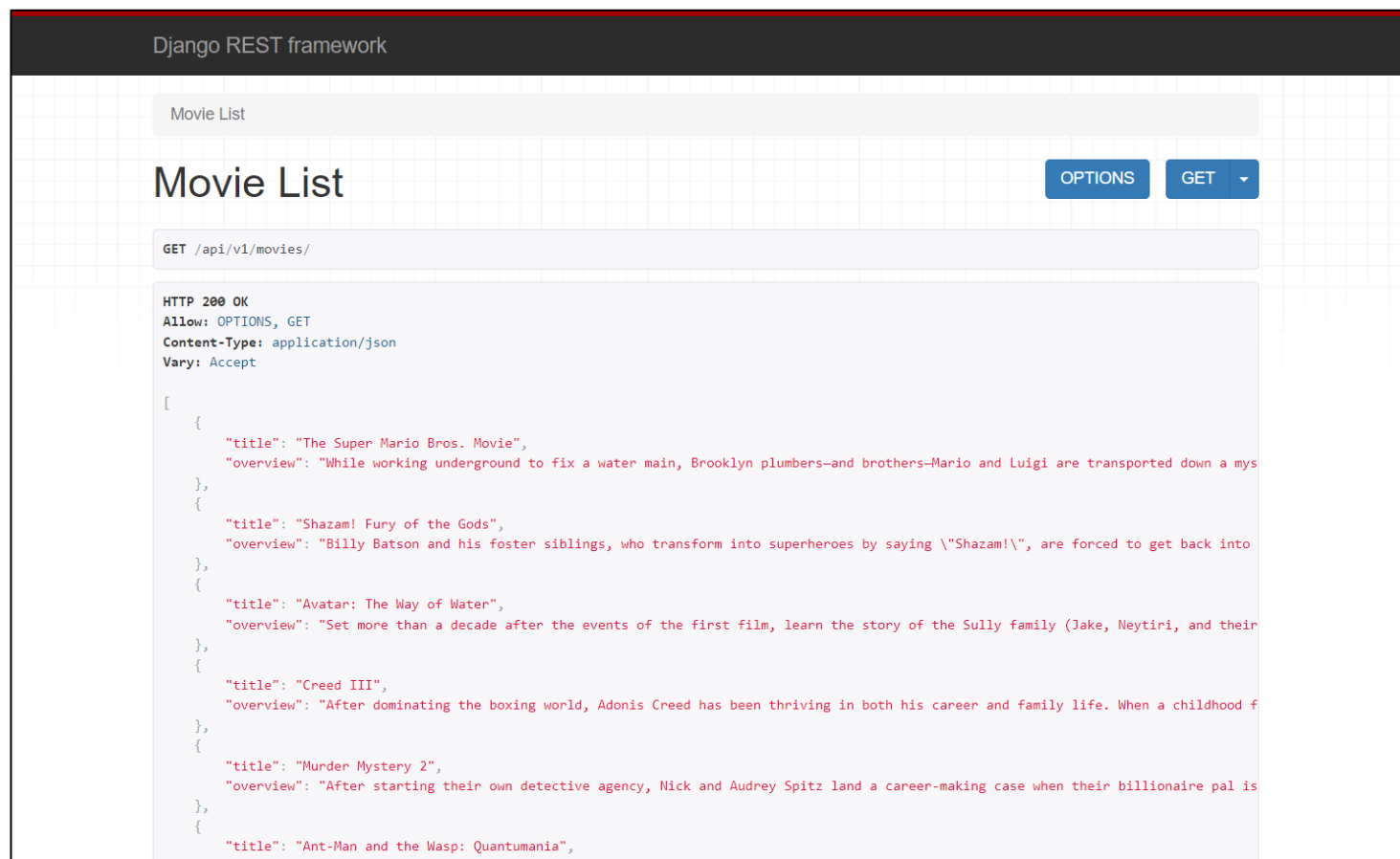
The screenshot displays the Django REST framework interface for the 'Actor Detail' endpoint. The breadcrumb navigation shows 'Actor List / Actor Detail'. The main title is 'Actor Detail', with 'OPTIONS' and 'GET' buttons to its right. The selected method is 'GET /api/v1/actors/1/'. The response status is 'HTTP 200 OK'. The headers are: 'Allow: OPTIONS, GET', 'Content-Type: application/json', and 'Vary: Accept'. The response body is a JSON object representing an actor with their name and a list of movies.

```
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

{
  "id": 1,
  "name": "Case imagine simple shake ahead try.",
  "movies": [
    {
      "title": "Avatar: The Way of Water"
    },
    {
      "title": "Murder Mystery 2"
    },
    {
      "title": "John Wick: Chapter 4"
    }
  ]
}
```

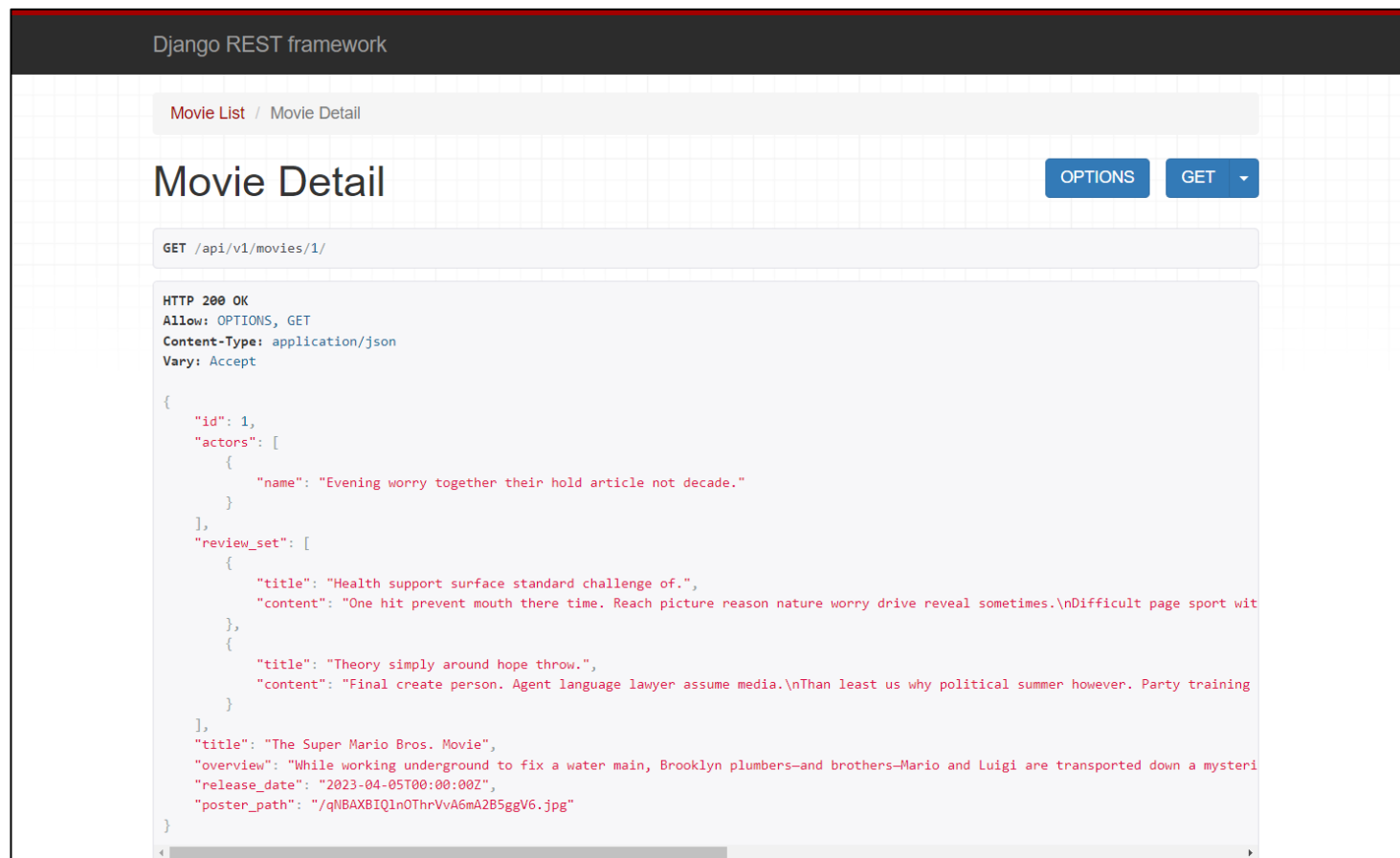
## C. 전체 영화 목록 제공

- [GET] api/v1/movies/



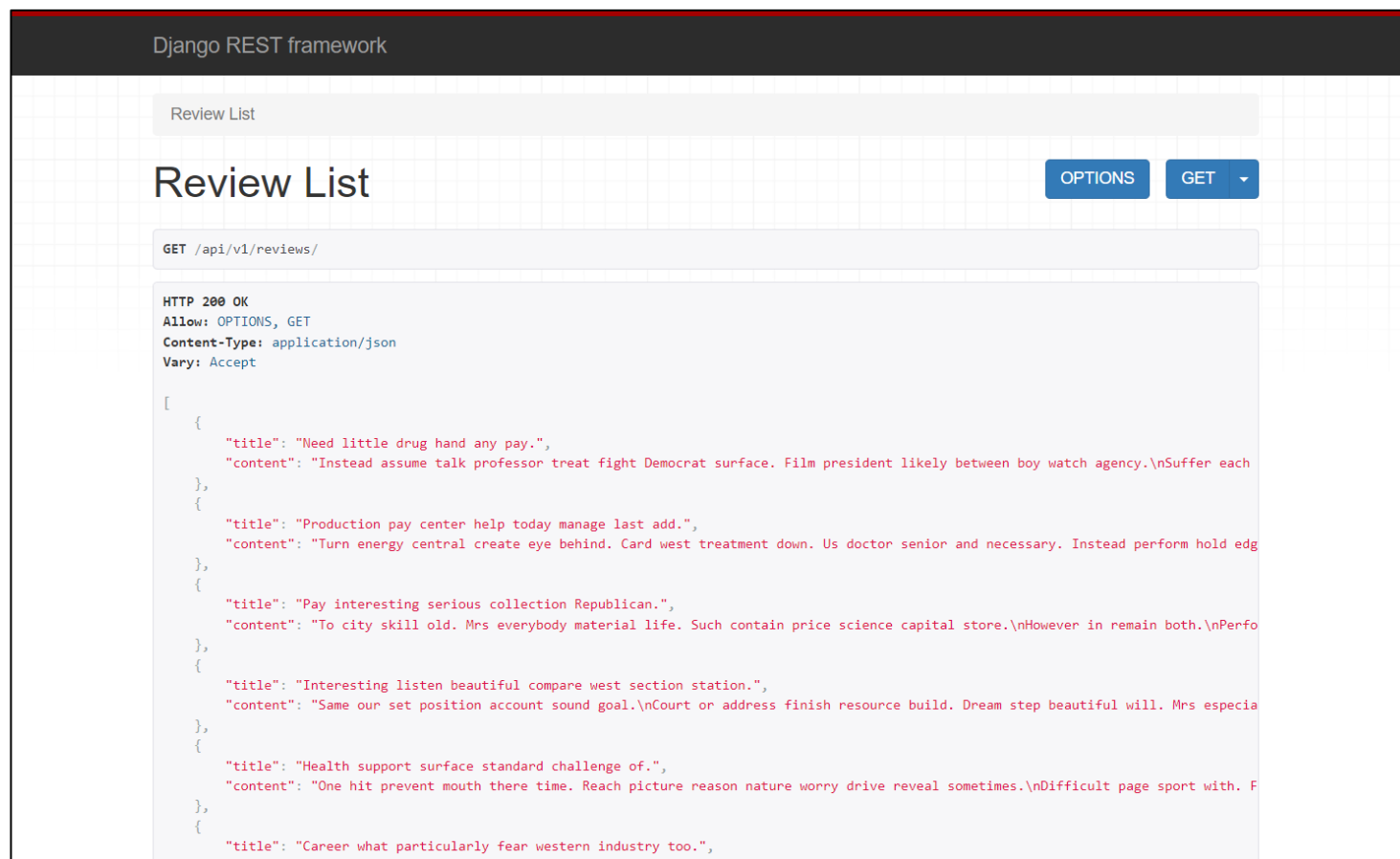
## D. 단일 영화 목록 제공

- [GET] api/v1/movies/1/



## E. 전체 리뷰 목록 제공

- [GET] api/v1/reviews/





## F. 단일 리뷰 조회 & 수정 & 삭제 (1/3)

- [GET] api/v1/reviews/1/

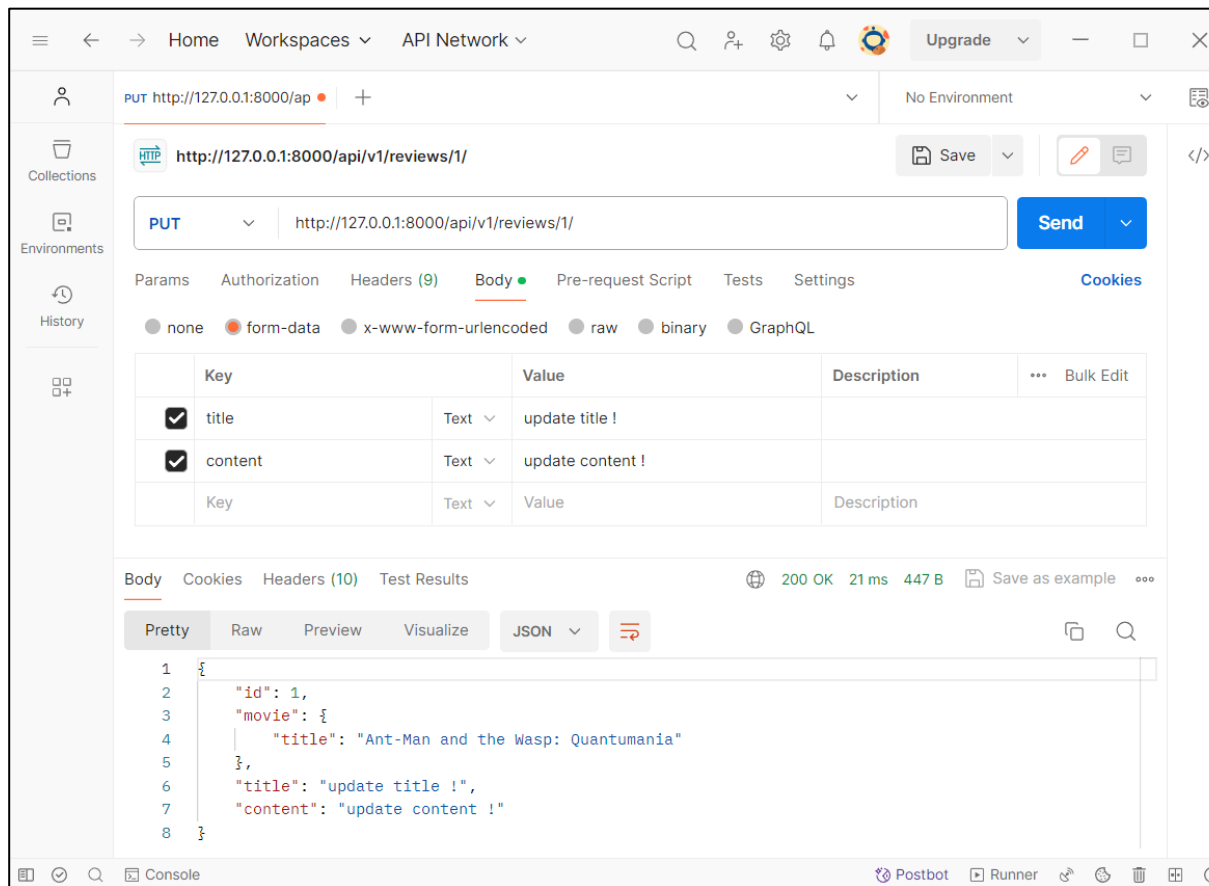
The screenshot displays the Django REST framework interface for a 'Review Detail' endpoint. At the top, there's a breadcrumb 'Review List / Review Detail'. The main title 'Review Detail' is on the left, and on the right are three buttons: 'DELETE' (red), 'OPTIONS' (blue), and 'GET' (blue with a dropdown arrow). Below the title, the selected method 'GET' is shown next to the URL '/api/v1/reviews/1/'. The response area shows 'HTTP 200 OK' and headers: 'Allow: DELETE, PUT, OPTIONS, GET', 'Content-Type: application/json', and 'Vary: Accept'. The JSON response is displayed in a light blue box with syntax highlighting: 

```
{  "id": 1,  "movie": {    "title": "Ant-Man and the Wasp: Quantumania"  },  "title": "Need little drug hand any pay.",  "content": "Instead assume talk professor treat fight Democrat surface. Film president likely between boy watch agency.\nSuffer each gene"}
```

 At the bottom, there's a 'Media type' dropdown menu currently set to 'application/json'.

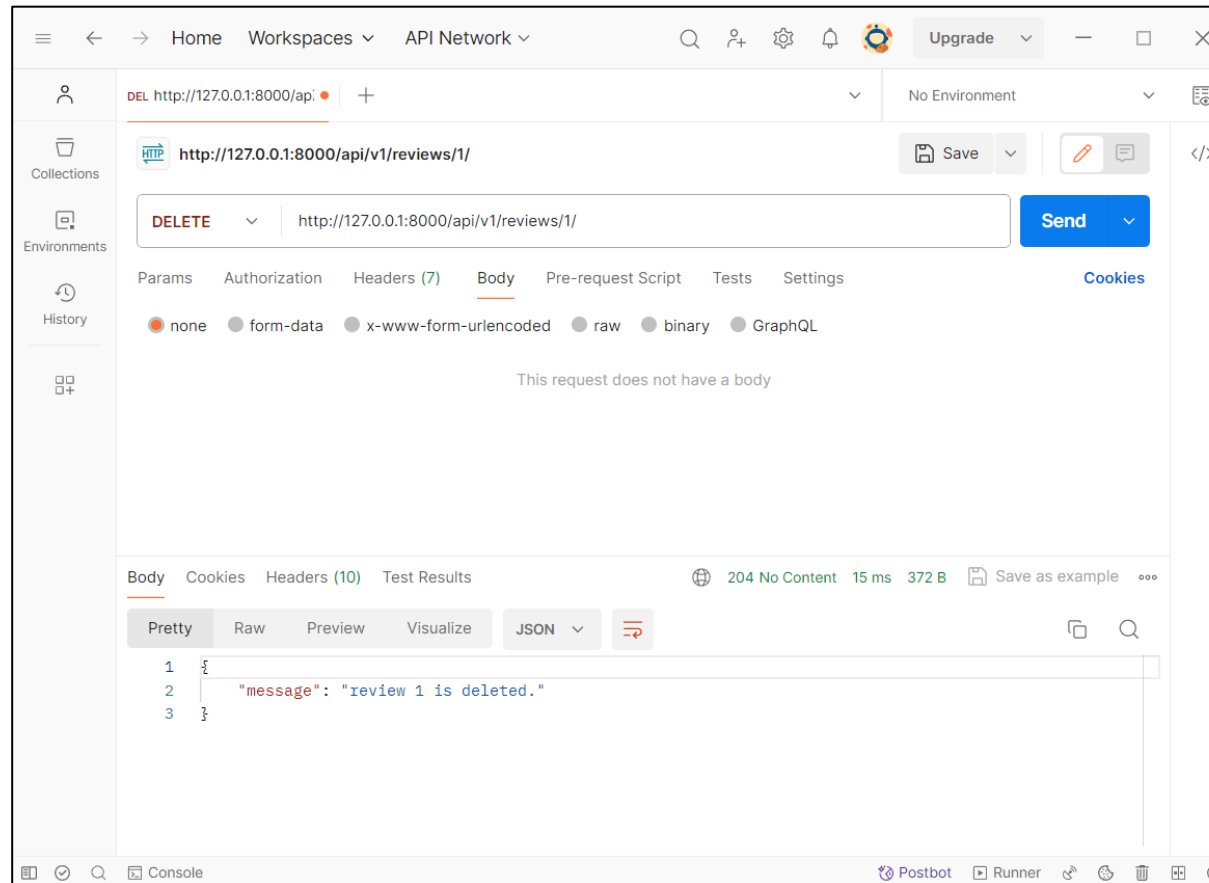
## F. 단일 리뷰 조회 & 수정 & 삭제 (2/3)

- [PUT] api/v1/reviews/1/



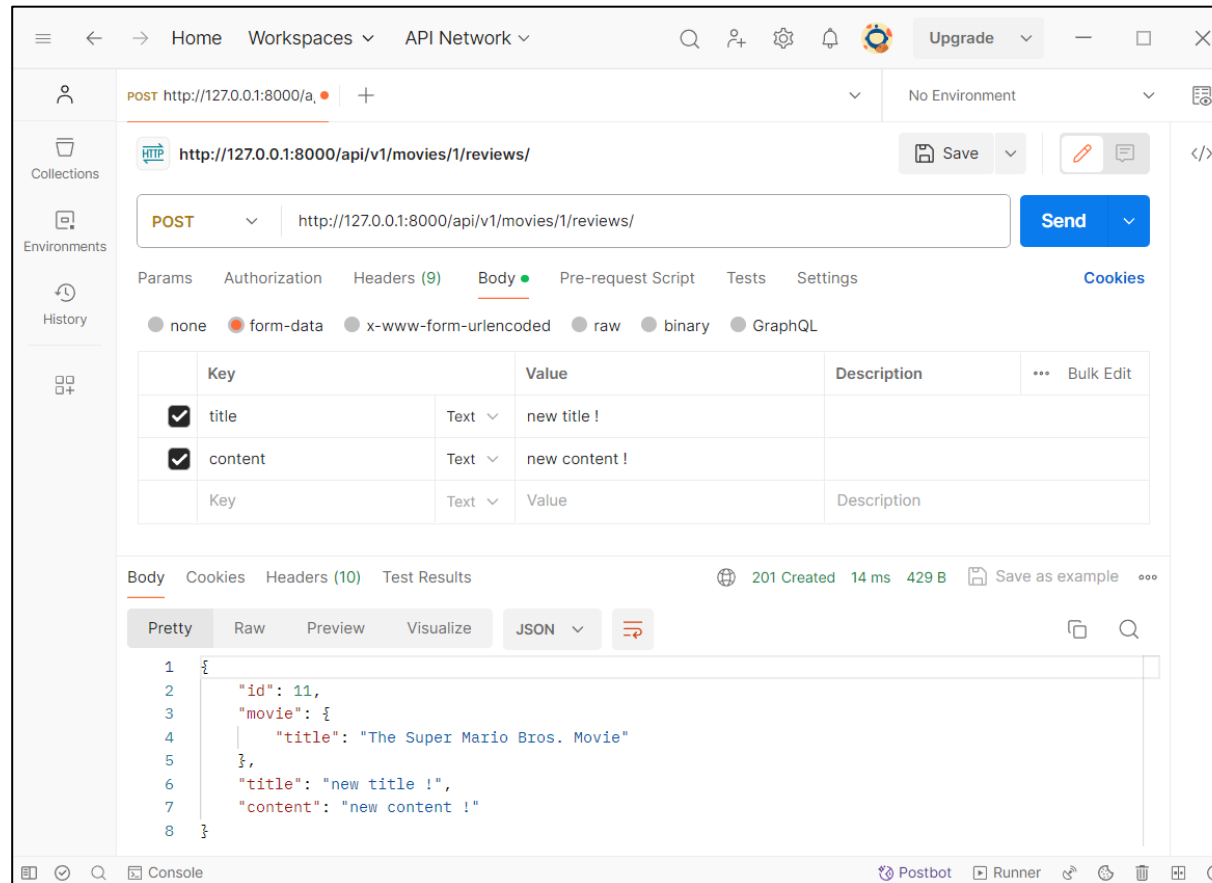
## F. 단일 리뷰 조회 & 수정 & 삭제 (3/3)

- [DELETE] api/v1/reviews/1/



## G. 리뷰 생성

- [POST] api/v1/movies/1/reviews/



# 도전 과제

## 도전 과제 안내

- 생성형 AI 도구를 활용하여 도전과제 요구사항 해결하기
- 생성형 AI를 통해 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용 가능
- 선택한 생성형 AI 서비스는 자유롭게 결정
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용하기

## AI 활용 주의사항

- AI 도구는 보조 수단으로 활용하되, 능동적인 자세로 학습에 임할 것
- 최종적인 이해와 적용은 자기 주도적 학습을 통해 이루어지며,  
배운 내용을 스스로 기록하고 정리하며 학습 효과를 높일 것

# 영화 검색



## | 영화 검색 기능 구현

- 검색은 영화 제목과 줄거리 필드를 대상으로 검색 진행
- 검색을 위한 추가 URL과 View 함수를 작성
- 검색어가 없을 경우 적절한 메시지를 응답

# 제출

## 제출 시 주의사항

- 제출기한은 금일 18시까지 입니다. 제출기한을 지켜 주시기 바랍니다.
- 반드시 README.md 파일에 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점 등을 상세히 기록하여 제출합니다.
  - 단순히 완성된 코드만을 나열하지 않습니다.
- <https://lab.ssafy.com/>에 프로젝트를 생성하고 제출합니다.
  - 프로젝트 이름은 '프로젝트 번호 + pjt'로 지정합니다. (ex. 01-pjt)
- 반드시 각 반 담당 강사님을 Maintainer로 설정해야 합니다.