

PJT명	Database 설계를 활용한 REST API 설계	
단계	[Django 개발]	
진행일자	2025.11.14	
예상 구현 시간	필수기능	5H
	추가기능	2H
	심화기능	1H

## 1. 목표

- Django REST Framework를 활용하여 API 서버를 제작할 수 있다.
- HTTP request methods에 대하여 이해한다.
- HTTP response status codes에 대하여 이해한다.
- Many to one relationship(N:1)에 대하여 이해한다.
- Many to many relationship(N:M)에 대하여 이해한다.

## 2. 준비사항

### 1) 제공 파일

- fixture의 형태로 제공되는 초기 데이터 (JSON 파일)
  - actors.json, movies.json, reviews.json

### 2) 개발언어 및 툴

- Python 3.11
- Postman
- Visual Studio Code

### 3) 필수 라이브러리 / 오픈소스

- Django 5.2
- Django REST Framework

### 3. 작업 순서

- 1) 팀원과 같이 요구사항을 확인하고, GitLab에 프로젝트를 생성한다.
  - 프로젝트 이름은 07-pjt로 지정한다.
  - 각 반 담당 강사님을 Maintainer로 설정한다.
- 2) 팀원과 합의하여 협업 방식을 결정하고, 요구사항을 구현한다.
- 3) 작성한 코드들을 정리하고, README를 작성한다.
  - .gitignore 파일을 활용하여 불필요한 파일 및 폴더는 제출하지 않는다.
- 4) README 작성이 완료되면 심화 학습을 진행한다.
- 5) 제출 기한에 맞춰 모든 산출물이 GitLab에 업로드 될 수 있도록 한다.

## 4. 요구사항

추천 알고리즘을 통한 영화 추천 커뮤니티 서비스를 구축하려고 한다. 다양한 스트리밍 플랫폼에서 제공되는 영화 정보를 수집 및 관리하고, 이를 기반으로 개인화된 영화 추천, 장르별 영화 탐색, 유사 영화 추천 등 다채로운 추천 기능을 설계 및 구현한다. 또한 영화에 대한 사용자 리뷰 및 감상평 공유 커뮤니티 기능을 제공하여, 사용자들이 활발하게 소통하고 정보를 교환할 수 있는 기능을 제공한다. 사용자는 자신이 본 영화를 평가하고, 다른 사용자의 리뷰를 참고하여 다음 영화를 선택하는데 도움을 받을 수 있다. 나아가, 관심 영화 목록을 맞춤형으로 구성하는 등 다양한 편의 기능을 제공한다. 팀원과 상의하여 아래 요구사항을 만족할 수 있도록 요구 사항 명세서를 작성 및 구현해보자.

영화 관련 데이터를 제공하는 RESTful API 서버를 만들고자 한다. 영화, 배우, 리뷰 등 서비스와 연관된 데이터의 조회가 가능하며, 사용자가 리뷰를 남기고 관리할 수 있는 API를 제공하는 서버를 구현해보자.

- 요구사항 예시(참고용)
  - 아래의 내용을 참고하여 추가적인 아이디어에 대해 요구사항을 추가 또는 수정하여 기능을 구현한다. 단, **필수 기능은 구현해야 하며, 수정할 수 없다.**

번호	분류	요구사항명	요구사항 상세	우선순위
<b>기능적 요구사항</b>				
F01	프로젝트	프로젝트 구성	영화 커뮤니티 서비스의 API 서버 구현을 위한 Django 프로젝트 및 앱 생성	필수
F02	movies model	Actor 클래스	배우 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F03	movies model	Movie 클래스	영화 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F04	movies model	Review 클래스	리뷰 데이터를 데이터베이스에 저장할 수 있도록 Django Model 클래스 구현	필수
F05	movies serializers	Serializer 클래스	사용자의 입력 데이터 검증 및 응답 데이터 형식을 위한 Serializer 클래스를 개별 요구사항에 맞춰서 구현	필수
F06	movies view	actor_list	전체 배우 데이터를 조회하는 view 함수 구현	필수
F07	movies view	actor_detail	단일 배우 데이터를 조회하는 view 함수 구현	필수
F08	movies view	movie_list	전체 영화 데이터를 조회하는 view 함수 구현	필수
F09	movies view	movie_detail	단일 영화 데이터를 조회하는 view 함수 구현	필수
F10	movies view	review_list	전체 리뷰 데이터를 조회하는 view 함수 구현	필수
F11	movies view	review_detail	단일 리뷰 데이터를 조회, 수정, 삭제하는 view 함수 구현	필수
F12	movies view	create_review	전달받은 리뷰 데이터를 데이터베이스에 저장하는 view 함수 구현	필수
F13	AI 활용	영화 검색	영화 제목과 줄거리를 대상으로 검색을 진행하는 기능 구현	도전
<b>비기능적 요구사항</b>				
NF01	프로젝트 관리	Git 활용	개발자 간 작업 충돌이 일어나지	필수

			않게끔 Git을 활용하여 프로젝트 관리	
NF02	설계	RESTful 원칙	RESTful 설계 원칙을 잘 따르도록 하여 사용성 증대	필수
NF03	보안	HTTP Method 허용	허용된 HTTP Method를 사용하는 요청만 허락하도록 구현	필수

## 1) 프로젝트 관리 (필수)

Git의 기능을 적극적으로 활용하여, 여러 개발자가 동시에 개발할 수 있는 환경을 조성한다.

- 요구사항 번호: NF01
- 팀원과 합의하여 branch 생성 원칙 정리하기
  - Gitflow, GitHub Flow 등을 참고하여 원칙을 정할 수 있음
  - branch 운영 중 문제상황이 발생할 경우 원인에 대한 분석을 꼭 진행할 것
- 팀원과 합의하여 commit 내역 기록 원칙 정리하기
  - 기능 개발, 버그 수정 등 그 목적이 명확히 드러날 수 있도록 commit 메시지를 작성할 것
  - 어느 시점에 commit을 남길지에 대하여 충분한 합의를 이를 것
- 프로젝트 종료 시 초기에 세운 원칙이 잘 지켜졌는지 점검
  - README에 초기에 세운 원칙을 설명하여 정리할 것
  - 생성했던 branch 들에 대한 용도를 정리할 것
  - 작성된 commit들을 GitLab에서 확인하여 스크린샷으로 README에 포함시킬 것

## 2) 프로젝트 및 앱 (필수)

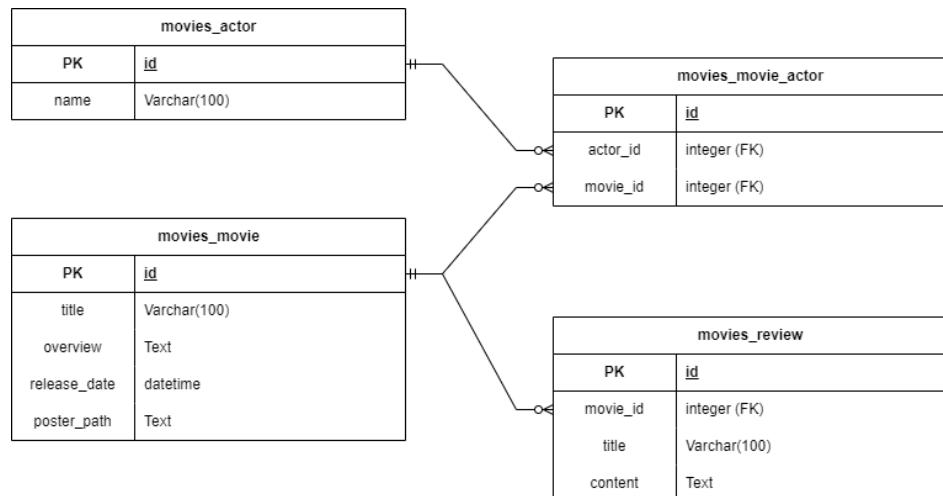
영화 데이터를 생성, 조회, 수정, 삭제할 수 있는 API를 제공하는 Django 프로젝트를 만든다. 명시된 요구사항 이외 서비스를 위해 필요하다고 생각되는 기능 등은 자유롭게 구현해도 무관하다.

- 요구사항 번호: F01
- 프로젝트 이름: mypjt
- 앱 이름: movies

## A. movies

영화 데이터를 관리하기 위한 앱이다. 요구사항으로 Model 클래스를 정의해보자.

- 요구사항 번호: F02, F03, F04, F05, NF02, NF03
- Actor 클래스
  - 배우 이름(name)을 저장할 필드 1개 지정.
- Movie 클래스
  - 영화 제목, 줄거리, 개봉일, 포스터 주소(title, overview, release\_date, poster\_path)를 저장할 필드 4개 지정
  - Actor 클래스와 N:M 관계를 맺고 있는, 출연 배우를 나타내는 필드(actors) 지정
- Review 클래스
  - 리뷰 제목, 내용(title, content)을 입력할 필드 2개 지정
  - Movie와 N:1 관계를 맺고 있음을 나타내는 필드(movie) 지정
- Model 클래스를 만든 후 제공된 fixture로 초기 데이터 불러오기
  - 각 Model 클래스의 관계에 주의할 것
- 기능 개발 시 유효성 검증을 위한 Serializer를 요구사항 구현에 필요한 형태로 작성하여 사용
- RESTful 설계 원칙에 알맞게 URL을 설계할 것
- 아래 ERD 참고



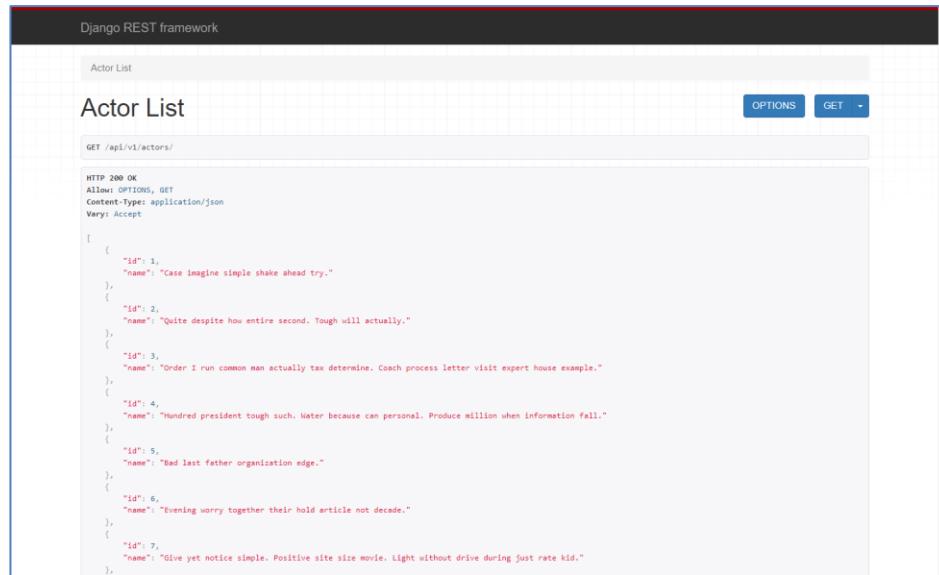
### 3) movies 앱 view 함수

영화 서비스와 연관된 다양한 데이터를 생성, 조회, 수정, 삭제하는 기능을 제공하는 movies 앱을 구현하자.

#### A. actor\_list

전체 배우 목록을 제공해주는 actor\_list view 함수를 구현한다.

- 요구사항 번호: F06
- 각 배우의 id, 이름 데이터가 조회되어야 함
- URL: /api/v1/actors/
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 HTML을 Django REST Framework가 제공하는 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



The screenshot shows a browser window with the title "Django REST framework". The main content area is titled "Actor List" and contains the following JSON response:

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

[
    {
        "id": 1,
        "name": "Case imagine simple shake ahead try."
    },
    {
        "id": 2,
        "name": "Quite despite how entire second. Tough will actually."
    },
    {
        "id": 3,
        "name": "Order I run common man actually tax determine. Coach process letter visit expert house example."
    },
    {
        "id": 4,
        "name": "Hundred president tough such. Water because can personal. Produce million when information fall."
    },
    {
        "id": 5,
        "name": "Bad last father organization edge."
    },
    {
        "id": 6,
        "name": "Evening worry together their hold article not decade."
    },
    {
        "id": 7,
        "name": "Give yet notice simple. Positive site size movie. Light without drive during just rate kid."
    }
]
```

## B. actor\_detail

단일 배우 정보를 제공해주는 actor\_detail view 함수를 구현한다.

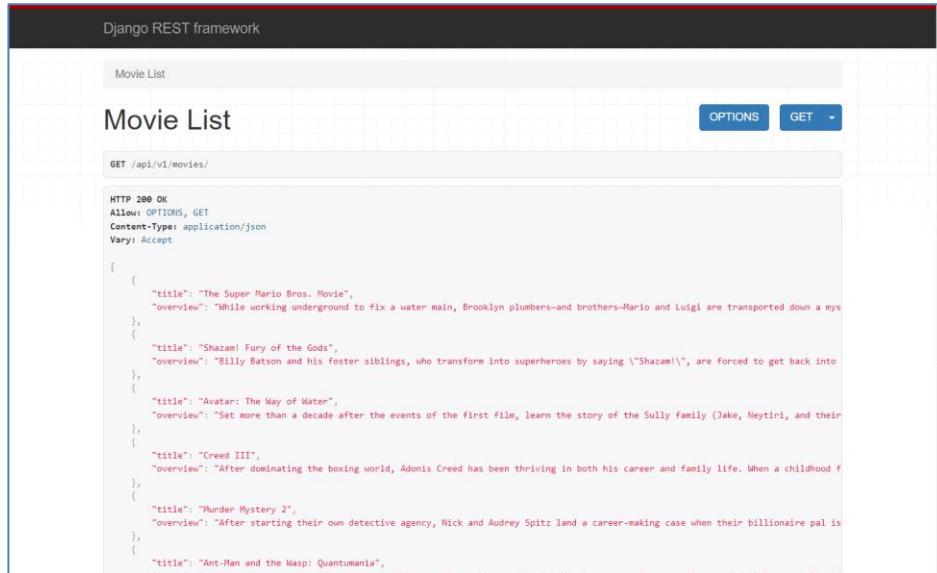
- 요구사항 번호: F07
- 해당하는 배우의 데이터와 함께, 출연한 영화의 제목이 목록으로 함께 조회되어야 함
- URL: /api/v1/actors/<배우ID>/
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시

```
Django REST framework
Actor List / Actor Detail
Actor Detail
GET /api/v1/actors/1/
OPTIONS GET
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. movie\_list

전체 영화 목록을 제공해주는 movie\_list view 함수를 구현한다.

- 요구사항 번호: F08
- 각 영화의 id, 제목, 줄거리 정보가 조회되어야 함
- URL: /api/v1/movies/
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시



The screenshot shows the Django REST framework's browsable API interface. At the top, it says "Movie List". Below that, there are two buttons: "OPTIONS" and "GET". The "GET" button is highlighted. Underneath the buttons, the URL "GET /api/v1/movies/" is shown. Below the URL, the response status is "HTTP 200 OK" and the content type is "application/json". The response body is a JSON array containing six movie objects. Each object has a "title" and an "overview". The titles are: "The Super Mario Bros. Movie", "Shazam! Fury of the Gods", "Avatar: The Way of Water", "Creed III", "Murder Mystery 2", and "Ant-Man and the Wasp: Quantumania". The overviews provide brief descriptions of each movie's plot.

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

[
    {
        "title": "The Super Mario Bros. Movie",
        "overview": "While working underground to fix a water main, Brooklyn plumbers-and brothers-Mario and Luigi are transported down a mys"
    },
    {
        "title": "Shazam! Fury of the Gods",
        "overview": "Billy Batson and his foster siblings, who transform into superheroes by saying \"Shazam!\", are forced to get back into"
    },
    {
        "title": "Avatar: The Way of Water",
        "overview": "Set more than a decade after the events of the first film, learn the story of the Sully family (Jake, Neytiri, and their"
    },
    {
        "title": "Creed III",
        "overview": "After dominating the boxing world, Adonis Creed has been thriving in both his career and family life. When a childhood f"
    },
    {
        "title": "Murder Mystery 2",
        "overview": "After starting their own detective agency, Nick and Audrey Spitz land a career-making case when their billionaire pal is"
    },
    {
        "title": "Ant-Man and the Wasp: Quantumania"
    }
]
```

## D. movie\_detail

단일 영화 정보를 제공해주는 movie\_detail view 함수를 구현한다

- 요구사항 번호: F09
- 해당하는 영화의 데이터와 함께, 영화에 출연한 배우 이름과 리뷰가 목록으로 함께 조회되어야 함
  - 조회되는 리뷰의 경우 제목, 내용 데이터가 조회되어야 함
- URL: /api/v1/movies/<영화ID>/
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시

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

{
    "id": 1,
    "actors": [
        {
            "name": "Evening worry together their hold article not decade."
        }
    ],
    "review_set": [
        {
            "title": "Health support surface standard challenge of.",
            "content": "One hit prevent mouth there time. Reach picture reason nature worry drive reveal sometimes.\nDifficult page sport wit"
        },
        {
            "title": "Theory simply around hope throw.",
            "content": "Final create person. Agent language lawyer assume media.\nThan least us why political summer however. Party training"
        }
    ],
    "title": "The Super Mario Bros. Movie",
    "overview": "While working underground to fix a water main, Brooklyn plumbers-and brothers-Mario and Luigi are transported down a mysterious hole.",
    "release_date": "2023-04-05T00:00:00Z",
    "poster_path": "/qjBAxBIQInOThrVvA6mz2B5ggV6.jpg"
}
```

## E. review\_list

전체 리뷰 목록을 제공해주는 review\_list view 함수를 구현한다.

- 요구사항 번호: F10
- 리뷰의 제목과 내용이 조회되어야 함
- URL: /api/v1/reviews/
- GET method에 대해서만 동작해야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- 브라우저 접근 출력 예시

```
Django REST framework
Review List
OPTIONS GET

GET /api/v1/reviews/

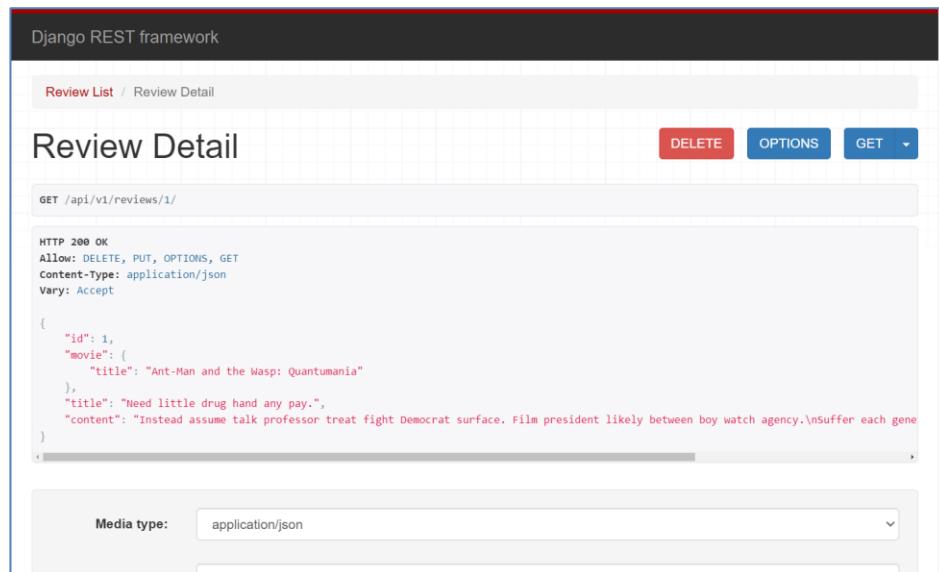
HTTP 200 OK
Allow: OPTIONS, GET
Content-Type: application/json
Vary: Accept

[
    {
        "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"
    },
    {
        "title": "Production pay center help today manage last add.",
        "content": "Turn energy central create eye behind. Card west treatment down. Us doctor senior and necessary. Instead perform hold edge"
    },
    {
        "title": "Pay interesting serious collection Republican.",
        "content": "To city skill old. Mrs everybody material life. Such contain price science capital store.\nHowever in remain both.\nPerfo"
    },
    {
        "title": "Interesting listen beautiful compare west section station.",
        "content": "Same our set position account sound goal.\nCourt or address finish resource build. Dream step beautiful will. Mrs especia"
    },
    {
        "title": "Health support surface standard challenge of.",
        "content": "One hit prevent mouth there time. Reach picture reason nature worry drive reveal sometimes.\nDifficult page sport with. F"
    },
    {
        "title": "Career what particularly fear western industry too."
    }
]
```

## F. review\_detail

단일 리뷰 정보를 request method에 따라 조회, 수정, 삭제해주는 review\_detail view 함수를 구현한다.

- 요구사항 번호: F11
- URL: /api/v1/reviews/<리뷰ID>/
- GET method로 요청했을 경우
  - 해당하는 리뷰 데이터와 함께, 대상 영화의 제목이 함께 조회되어야 함
  - 브라우저에서 요청했을 경우 Django REST Framework가 제공하는 HTML을 반환
  - Postman같은 API 클라이언트에서 요청했을 경우 JSON을 반환
- PUT method로 요청했을 경우
  - 유효한 데이터인 경우 대상 리뷰 수정
- DELETE method로 요청했을 경우
  - 대상 리뷰 삭제
- 브라우저 접근 출력 예시



The screenshot shows the Django REST framework's "Review Detail" view. At the top, there's a navigation bar with "Review List" and "Review Detail". On the right side of the detail page, there are three buttons: "DELETE" (red), "OPTIONS" (blue), and "GET" (blue). Below these buttons, the URL "GET /api/v1/reviews/1/" is displayed. The main content area shows the JSON response for the GET request:

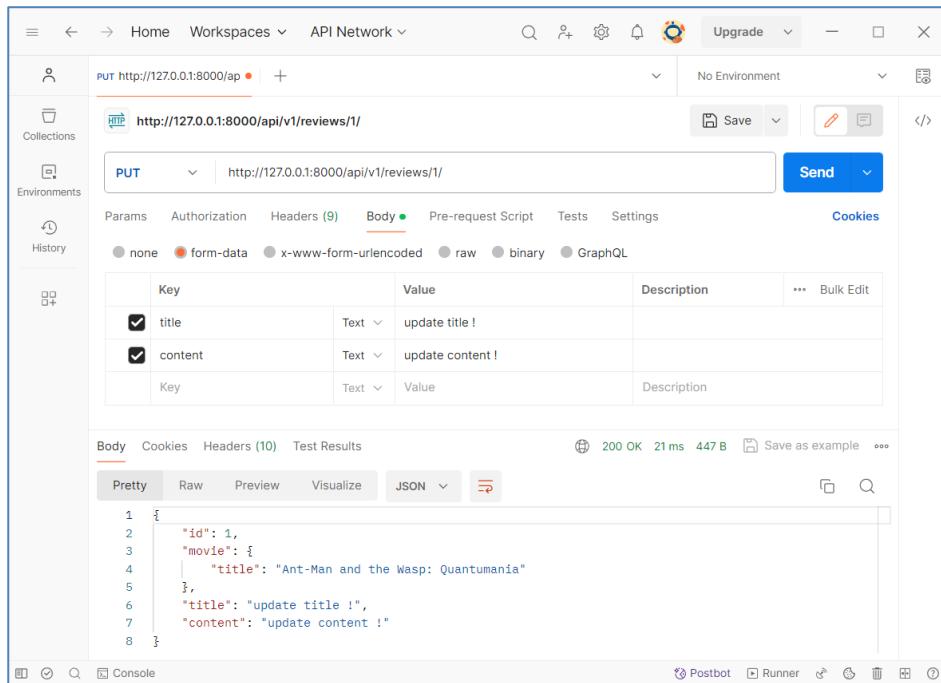
```
HTTP 200 OK
Allow: DELETE, PUT, OPTIONS, GET
Content-Type: application/json
Vary: Accept

{
    "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 of the screen, there's a "Media type:" dropdown menu set to "application/json".

## ● Postman 요청 예시

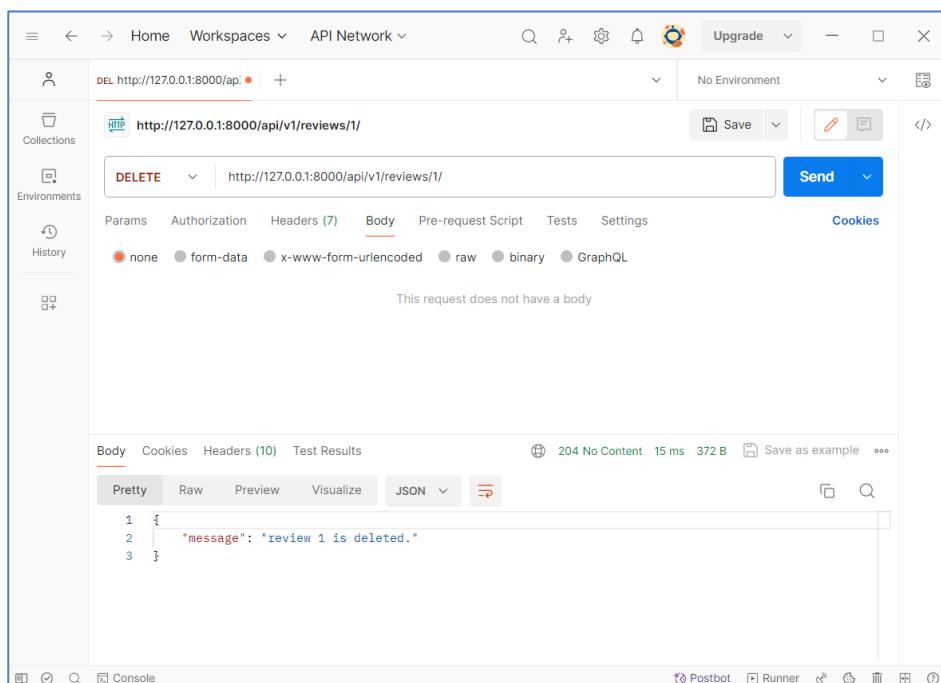
### - PUT method



The screenshot shows the Postman interface for a PUT request. The URL is `http://127.0.0.1:8000/api/v1/reviews/1`. The request type is `PUT`. The Body tab is selected, showing a JSON payload:

```
1 {  
2   "id": 1,  
3   "movie": {  
4     "title": "Ant-Man and the Wasp: Quantumania"  
5   },  
6   "title": "update title !",  
7   "content": "update content !"  
8 }
```

### - DELETE method



The screenshot shows the Postman interface for a DELETE request. The URL is `http://127.0.0.1:8000/api/v1/reviews/1`. The request type is `DELETE`. The Body tab is selected, showing the message: "This request does not have a body". The response message in the Body tab is: "message": "review 1 is deleted."

## G. create\_review

리뷰 데이터를 전달받아서 저장하는 create\_review view 함수를 구현한다.

- 요구사항 번호: F12
- URL: /api/v1/movies/<영화ID>/reviews/
- POST method에 대해서만 정상 동작해야 함
  - 제목, 내용 데이터를 전달받아 새로운 리뷰 생성
  - 유효한 데이터인 경우에만 정상 작동
- 그 외의 method의 경우 405 응답을 반환해야 함
  - 브라우저에서 접근할 경우 데이터를 입력할 수 있는 UI를 표시할 수 있음

### ● Postman 요청 예시

The screenshot shows the Postman interface with the following details:

- Request URL:** POST http://127.0.0.1:8000/api/v1/movies/1/reviews/
- Method:** POST
- Body (form-data):**

Key	Value	Description
title	new title !	
content	new content !	
- Response:**
  - Status: 201 Created
  - Time: 14 ms
  - Size: 429 B
  - Content (Pretty):

```
1 {  
2   "id": 11,  
3   "movie": {  
4     "title": "The Super Mario Bros. Movie"  
5   },  
6   "title": "new title !",  
7   "content": "new content !"  
8 }
```

## 4) 도전 과제

생성형 AI 도구를 활용하여 도전과제 요구사항을 해결해보자.

- 코드 생성, 아이디어 구상, 문제 해결 방법 탐색 등 다양한 방식으로 활용할 수 있다.
- 사용할 생성형 AI 서비스는 자유롭게 선택한다.
  - 제공되는 GPT API Key를 활용할 경우, 모델은 반드시 gpt-5-nano 모델을 사용한다.
- 최종 결과물은 AI 생성 내용을 바탕으로 직접 수정 및 개선하여 적용해본다.

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.

## A. 영화 검색 기능 구현

찾고 싶은 영화 정보를 검색할 수 있는 기능을 구현해보자.

- 요구사항 번호: F13
- 영화 제목과 줄거리 필드를 대상으로 검색 진행
- 검색을 위한 추가 URL과 view 함수를 작성
  - URL의 각 요소의 목적을 이해하고 활용할 것
- 검색어가 없을 경우 적절한 메시지를 응답할 것

## 5. 참고자료

- Django 5.2  
<https://www.djangoproject.com/start/overview/>
- Django REST Framework  
<https://www.django-rest-framework.org/>

## 6. 결과

제출 기한은 진행일 18시까지이므로 제출 기한을 지킬 수 있도록 한다. 제출은 GitLab을 통해서 이뤄진다.

### ● 산출물과 제출

- 단계별로 구현 과정 중 학습한 내용, 어려웠던 부분, 새로 배운 것들 및 느낀 점을 상세히 기록한 README.md
- 완성된 각 문제 별 소스코드 및 실행 화면 캡쳐본
- 프로젝트 이름은 07-pjt로 지정 및 각자 제출
  - 한 명의 GitLab 계정에 Git 저장소 Push 및 작업
  - 나머지 인원은 제출 시 해당 저장소를 Fork 하여 제출
- 각 반 담당 강사님을 Maintainer로 설정

- 끝 -