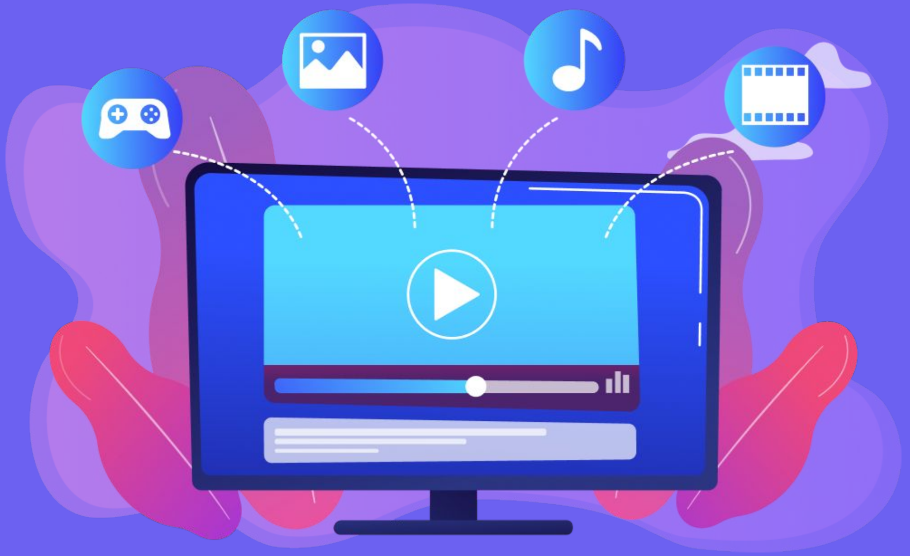
Build The Theatre-Movie Relationship Related APIs

Relevel
by Unacademy



Build the Theatre-Movie relationship related APIs

When we browse through any movie booking application we can search for all the movies running inside a theatre or all list all the theatres running the movie we want to see. In this lesson we will define a relation between theatre and movie and try to create endpoints that can show above results and more.

Session Agenda

- This session is associated with forming a relation between movie and theatre model.
- Here, we will first update theatre model to have a array of movie ids which will represent list of movies running in a theatre.
- We will then create some meaningful endpoints from users perspective.
- We will all create some endpoints that will be used by admins only to maintain list of movies inside a theatre.

From the previous class:

- So far, in our Movie Booking Application we have created our CRUD endpoints on Movie and Theatre by creating corresponding models, controller and routes using mongoDB and express and tested the same using Postman.
- In this class we will create some more endpoints which are some most common end points required to help user search any movie or theatre running any movie for e.g., to get all theatres where any specific movie is running or get list of movies running inside a theatre and so on.

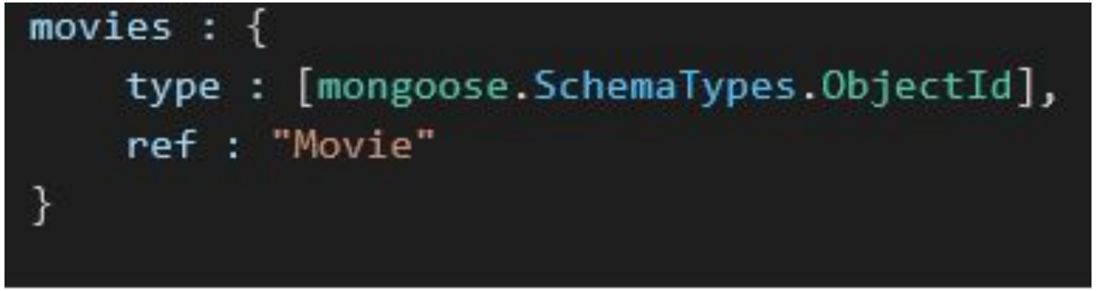
List of Concepts Involved

In this class, you will learn the following items:

1. Update Theatre model with movies attribute.
2. Add the movies inside a theatre.
3. Remove the movie inside the theatre.
4. Get the list of theatres in which a movie is running.
5. Search if a movie is running in any specific theatres.

Update Theatre model with movies attribute

- Before creating APIs that involves Theatres and Movies models, we need to have a relation between them
- This is implemented by adding an attribute named movies in Theatre model.
- It has a type as array of movie ids, which represents list of movies running inside a theatre.



```
movies : {
    type : [mongoose.SchemaTypes.ObjectId],
    ref : "Movie"
}
```

- After, this we need to update our init function in server.js file that creates some initial data in Theatre model by adding newly created attribute movies as shown below mentioned code link:

  https://github.com/Vishwa07dev/mba_backend/blob/session3/server.js

Now we can create endpoints that involves Theatre and Movie models together, let's go through them one by one.

Add the movies inside a theatre.

- This API will be used to add a movie inside a theatre
- We send a request body consisting of an array of movie ids named "movieIds"
- And another attribute named "insert" which will be a boolean.
- According to which either the insertion or removal of movie or movies will be done from a theatre model.
- Here, we will pass **true** value for insert attribute.
- Theatre model is obtained by the id passed inside the API route named as ":id".

- **API endpoint:**

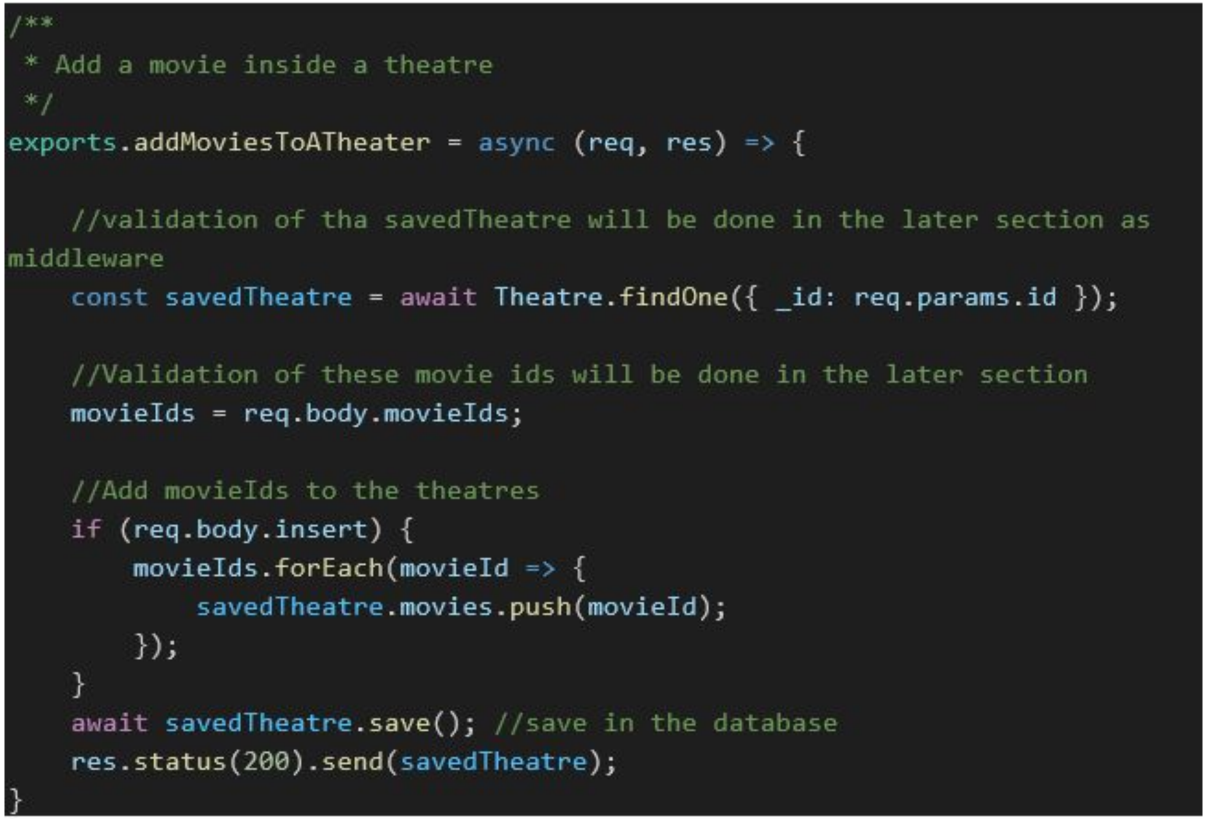
PUT /mba/api/v1/theatres/:id/movies

- **Creating a new function in Controller:**

Inside this function we will first be finding one theatre using the id params from the request and then update the movies array of the theatre model received from the findOne method after checking insert value from the request to be true and finally save the model to update it.

- **Code Link:**

https://github.com/Vishwa07dev/mba_backend/blob/session3/controllers/theatre.controller.js



```
/**
 * Add a movie inside a theatre
 */
exports.addMoviesToATheater = async (req, res) => {

    //validation of tha savedTheatre will be done in the later section as
middleware
    const savedTheatre = await Theatre.findOne({ _id: req.params.id });

    //Validation of these movie ids will be done in the later section
    movieIds = req.body.movieIds;

    //Add movieIds to the theatres
    if (req.body.insert) {
        movieIds.forEach(movieId => {
            savedTheatre.movies.push(movieId);
        });
    }
    await savedTheatre.save(); //save in the database
    res.status(200).send(savedTheatre);
}
```

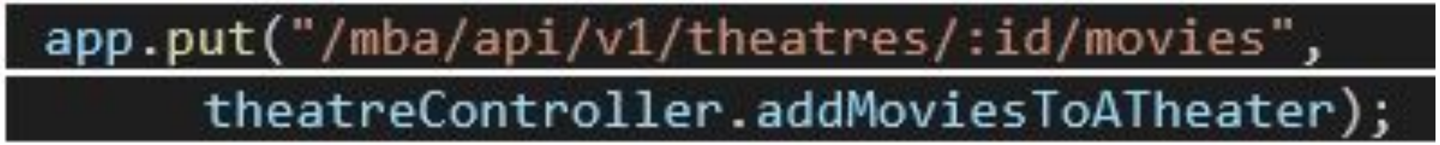
- **Creating new route:**

A new PUT route will be added inside theatre.routes.js file which will call the above created function **addMoviesToATheatre**.

We are using PUT route as this route is eventually updating a theatre document inside our database.

- **Code Link:**

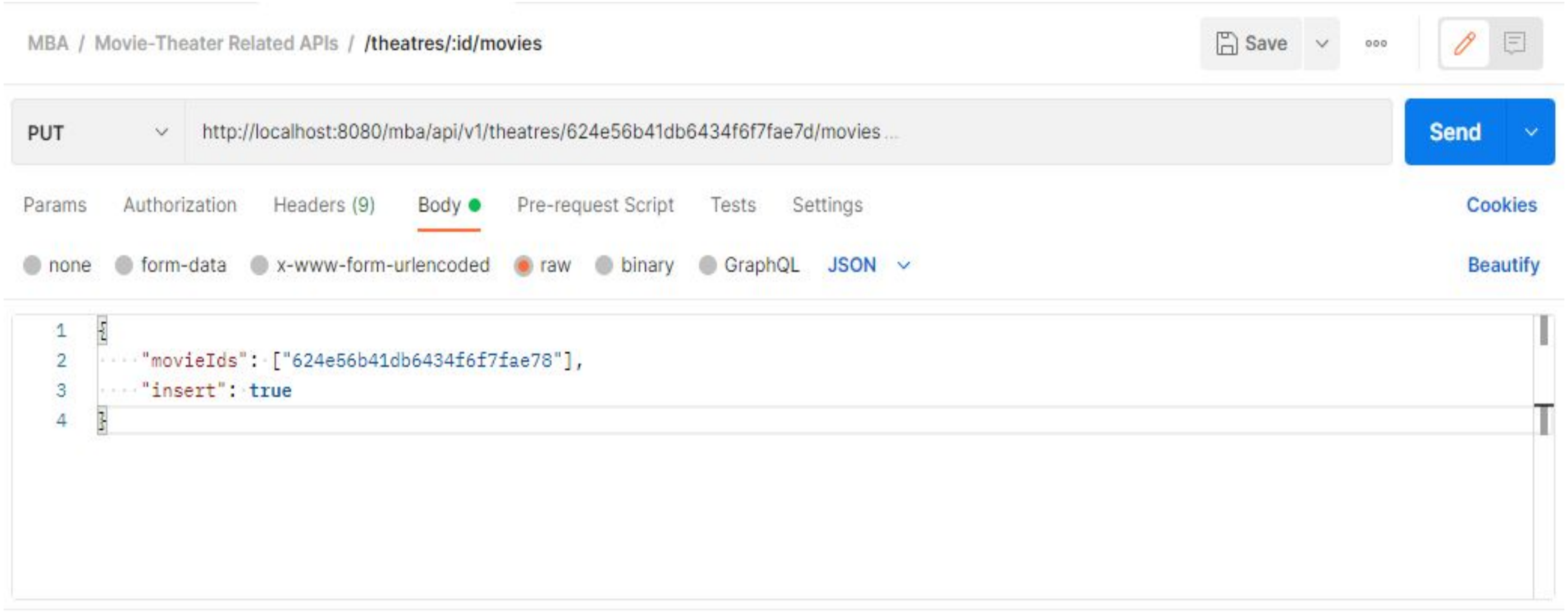
https://github.com/Vishwa07dev/mba_backend/blob/session3/routes/theatre.routes.js



```
app.put("/mba/api/v1/theatres/:id/movies",
        theatreController.addMoviesToATheater);
```

Testing the API:

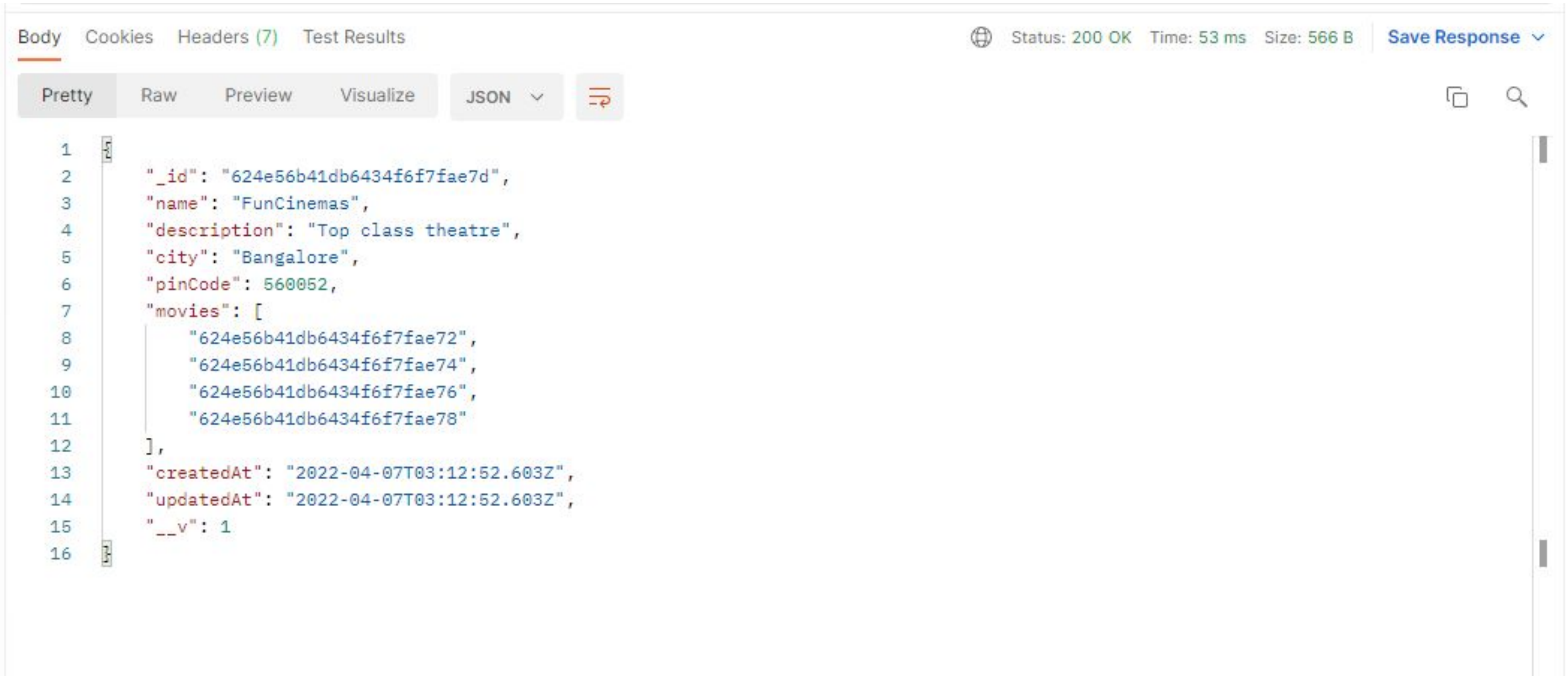
Request:


MBA / Movie-Theater Related APIs / /theatres/:id/movies

PUT http://localhost:8080/mba/api/v1/theatres/624e56b41db6434f6f7fae7d/movies ...

```
{
    "movieIds": ["624e56b41db6434f6f7fae78"],
    "insert": true
}
```

Response:



```
{
    "_id": "624e56b41db6434f6f7fae7d",
    "name": "FunCinemas",
    "description": "Top class theatre",
    "city": "Bangalore",
    "pinCode": 560052,
    "movies": [
        "624e56b41db6434f6f7fae72",
        "624e56b41db6434f6f7fae74",
        "624e56b41db6434f6f7fae76",
        "624e56b41db6434f6f7fae78"
    ],
    "createdAt": "2022-04-07T03:12:52.603Z",
    "updatedAt": "2022-04-07T03:12:52.603Z",
    "__v": 1
}
```

Remove the movie inside the theatre.

- This API will be used to remove a movie inside a theatre,
- We send a request body consisting of an array named “movieIds”
- and another attribute named “insert” which will be a Boolean according to which either the insertion or removal of movie or movies will be done from a theatre model.
- Here, we will pass **false** value for insert attribute.
- Theatre model is obtained by the id passed inside the API route named as “:id”.

- **API endpoint:**

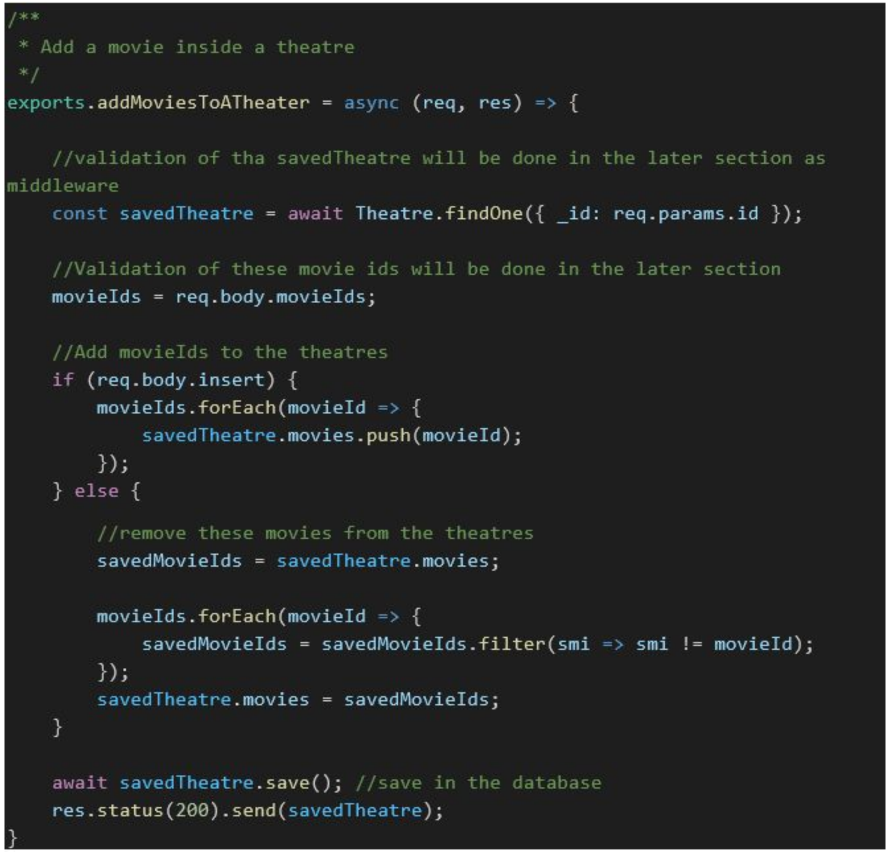
PUT /mba/api/v1/theatres/:id/movies

- **Using existing function in Controller:**

Here we are going to use the same function from controller named **addMoviesToATheater** but add an else condition on request body's "insert" attribute which when not be true then list of movies in the movieIds array from request body will be removed the Theatre model and the same will be updated.

- **Code Link:**

https://github.com/Vishwa07dev/mba_backend/blob/session3/controllers/theatre.controller.js



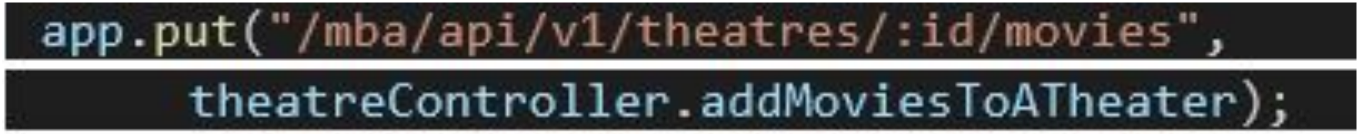
```
/**
 * Add a movie inside a theatre
 */
exports.addMoviesToATheater = async (req, res) => {

    //validation of tha savedTheatre will be done in the later section as
middleware
    const savedTheatre = await Theatre.findOne({ _id: req.params.id });

    //Validation of these movie ids will be done in the later section
    movieIds = req.body.movieIds;

    //Add movieIds to the theatres
    if (req.body.insert) {
        movieIds.forEach(movieId => {
            savedTheatre.movies.push(movieId);
        });
    } else {

        //remove these movies from the theatres
        savedMovieIds = savedTheatre.movies;

        movieIds.forEach(movieId => {
            savedMovieIds = savedMovieIds.filter(smi => smi != movieId);
        });
        savedTheatre.movies = savedMovieIds;
    }

    await savedTheatre.save(); //save in the database
    res.status(200).send(savedTheatre);
}
```

- **Using same route:**

We are using PUT route as this route is eventually updating a theatre document inside our database.

- **Code Link:**

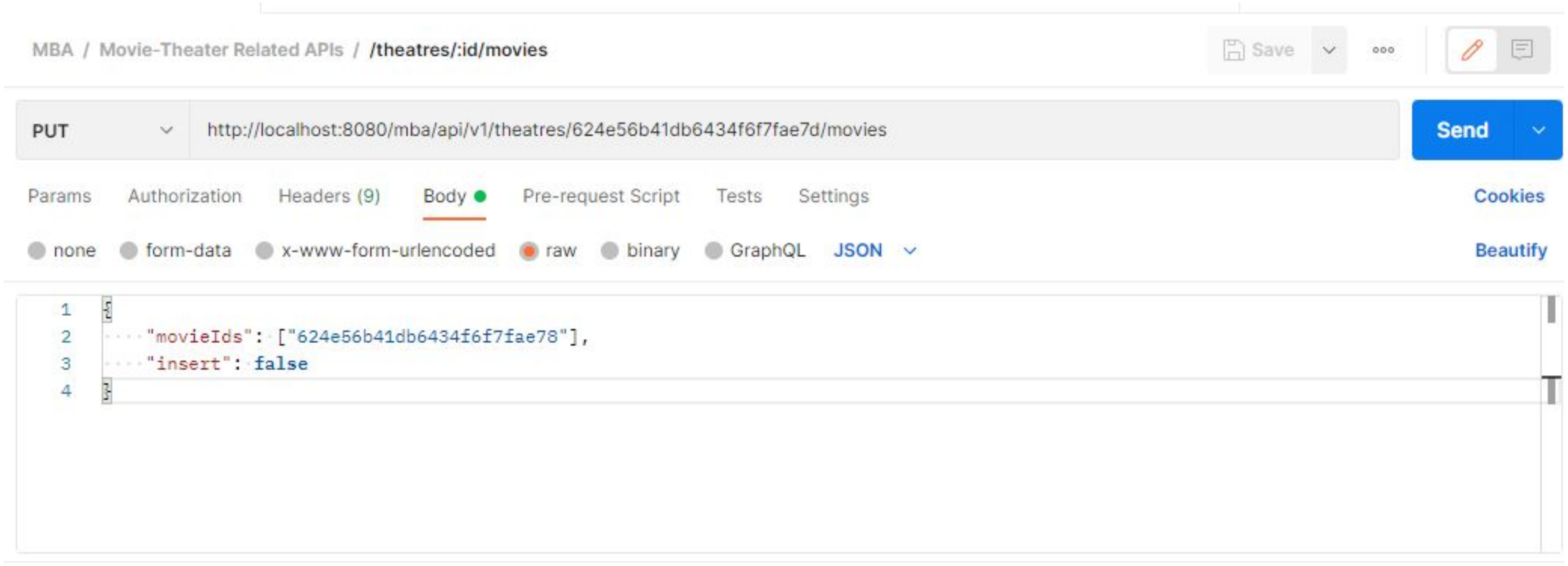
https://github.com/Vishwa07dev/mba_backend/blob/session3/routes/theatre.routes.js



```
app.put("/mba/api/v1/theatres/:id/movies",
    theatreController.addMoviesToATheater);
```

Testing the API:

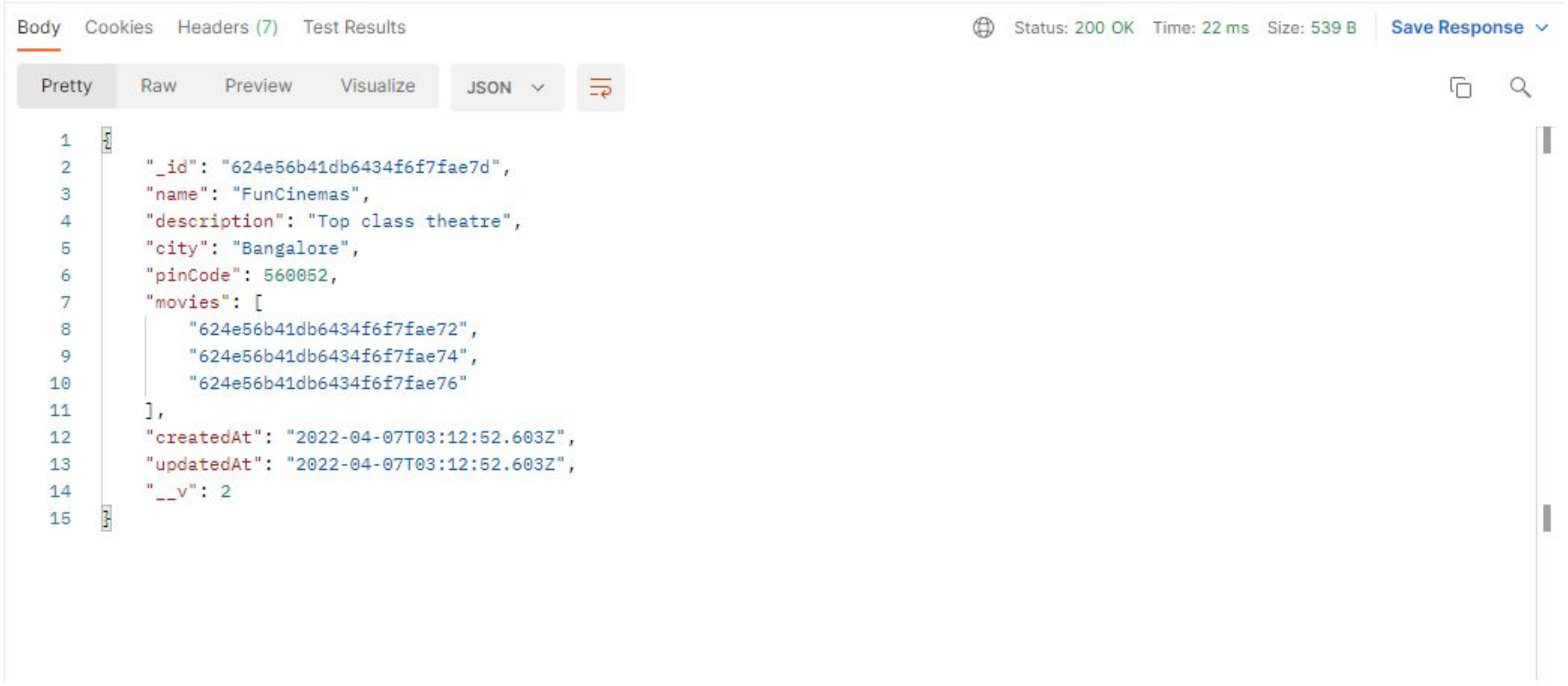
Request:


MBA / Movie-Theater Related APIs / /theatres/:id/movies

PUT http://localhost:8080/mba/api/v1/theatres/624e56b41db6434f6f7fae7d/movies

```
{
    "movieIds": ["624e56b41db6434f6f7fae78"],
    "insert": false
}
```

Response:



```
{
    "_id": "624e56b41db6434f6f7fae7d",
    "name": "FunCinemas",
    "description": "Top class theatre",
    "city": "Bangalore",
    "pinCode": 560052,
    "movies": [
        "624e56b41db6434f6f7fae72",
        "624e56b41db6434f6f7fae74",
        "624e56b41db6434f6f7fae76"
    ],
    "createdAt": "2022-04-07T03:12:52.603Z",
    "updatedAt": "2022-04-07T03:12:52.603Z",
    "__v": 2
}
```

Get the list of theatres in which a movie is running.

- This API will be used to get the list of all theatres in which a given movie is running.
- We do so by passing a query object in the request named as “movieId”.
- The response will contain only those theatre which has this provided movie running inside of it.
  - **API endpoint:**

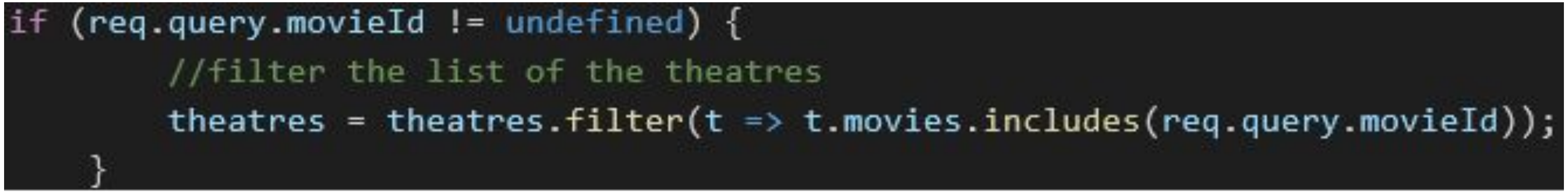
  GET /mba/api/v1/theatres?movieId=any_movie_id

  - **Using existing function in Controller:**

Here we are going to use the existing function from controller named **getAllTheatres** but add a conditional block at the bottom before sending the response to check if the request query has a value for attribute “movieId”. If the value exist we will then filter the theatres to only those which has the movie id in their movies array.

  - **Code Link:**

https://github.com/Vishwa07dev/mba_backend/blob/session3/controllers/theatre.controller.js



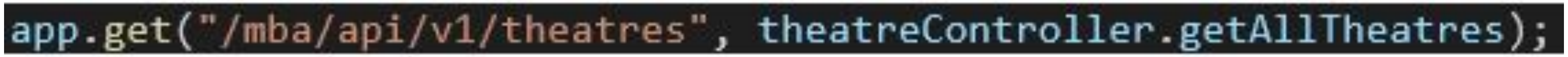
```
if (req.query.movieId != undefined) {
        //filter the list of the theatres
        theatres = theatres.filter(t => t.movies.includes(req.query.movieId));
    }
```

- **Route:**

Here we are going to use the same route itself.

- **Code Link:**

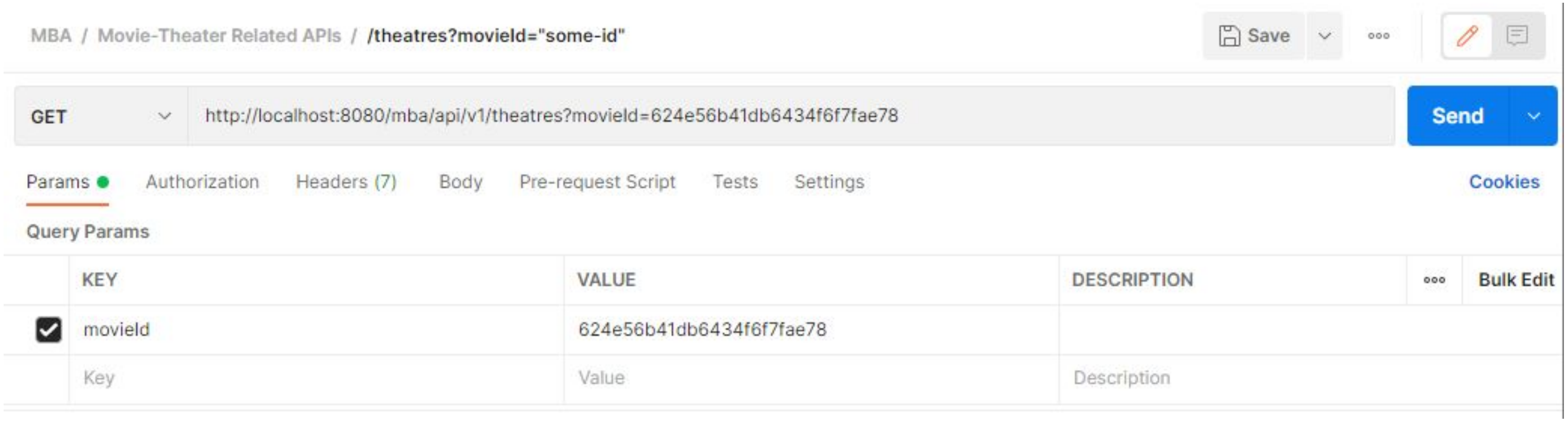
https://github.com/Vishwa07dev/mba_backend/blob/session3/routes/theatre.routes.js



```
app.get("/mba/api/v1/theatres", theatreController.getAllTheatres);
```

Testing the API:

Request:



MBA / Movie-Theater Related APIs / /theatres?movieId="some-id"

Save ooo

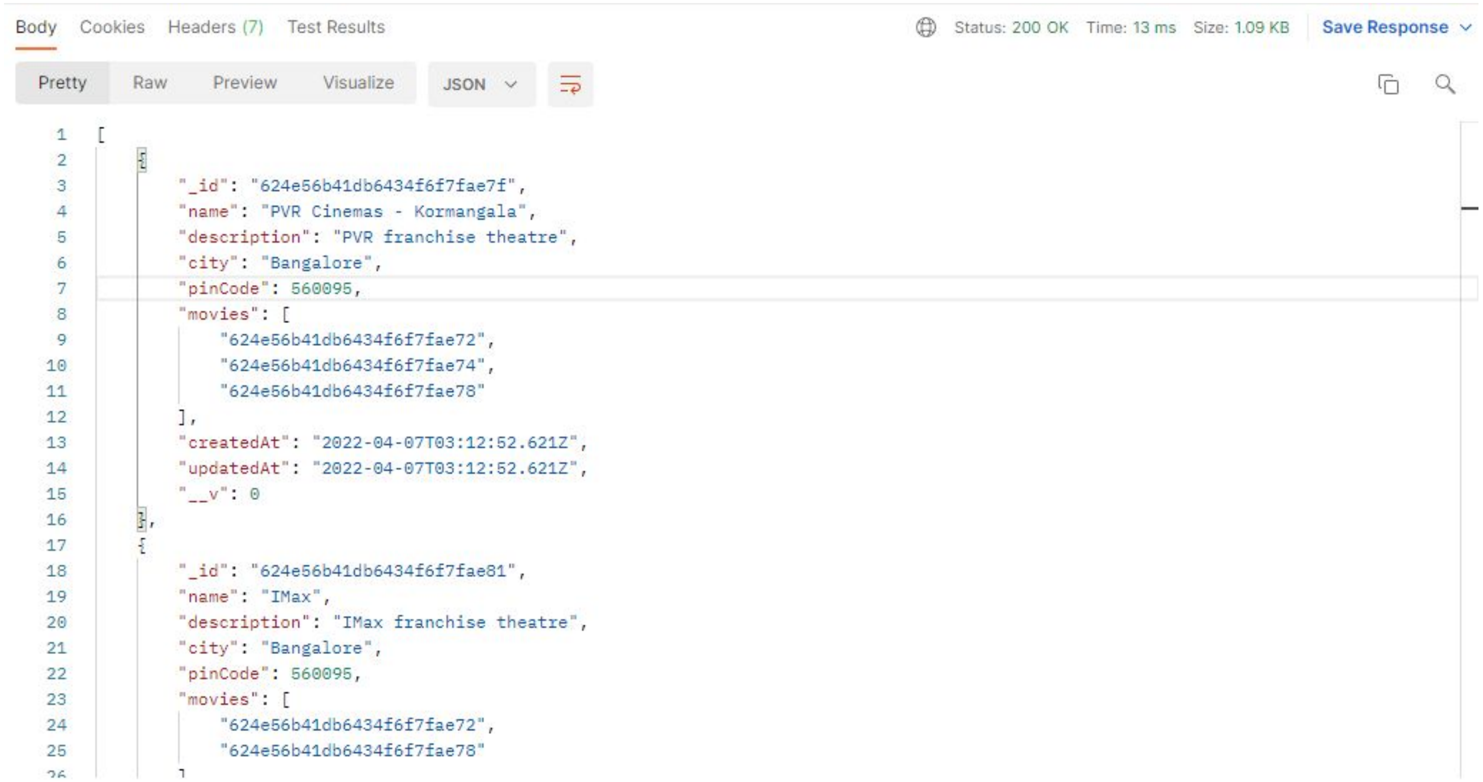
GET http://localhost:8080/mba/api/v1/theatres?movieId=624e56b41db6434f6f7fae78 Send

Params ● Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION
✓	movieId	624e56b41db6434f6f7fae78	
	Key	Value	Description

Response:


Body Cookies Headers (7) Test Results

Status: 200 OK Time: 13 ms Size: 1.09 KB

```
[
    {
        "_id": "624e56b41db6434f6f7fae7f",
        "name": "PVR Cinemas - Kormangala",
        "description": "PVR franchise theatre",
        "city": "Bangalore",
        "pinCode": 560095,
        "movies": [
            "624e56b41db6434f6f7fae72",
            "624e56b41db6434f6f7fae74",
            "624e56b41db6434f6f7fae78"
        ],
        "createdAt": "2022-04-07T03:12:52.621Z",
        "updatedAt": "2022-04-07T03:12:52.621Z",
        "__v": 0
    },
    {
        "_id": "624e56b41db6434f6f7fae81",
        "name": "IMax",
        "description": "IMax franchise theatre",
        "city": "Bangalore",
        "pinCode": 560095,
        "movies": [
            "624e56b41db6434f6f7fae72",
            "624e56b41db6434f6f7fae78"
        ]
```

Search if a movie is running in any specific theatres.

- This API will be used to check whether a specific movie is running inside a specific theatre.
- We get the detail of movie and the theatre from the request path itself named as “movieId” and “theatreId”.

  - **API endpoint:**

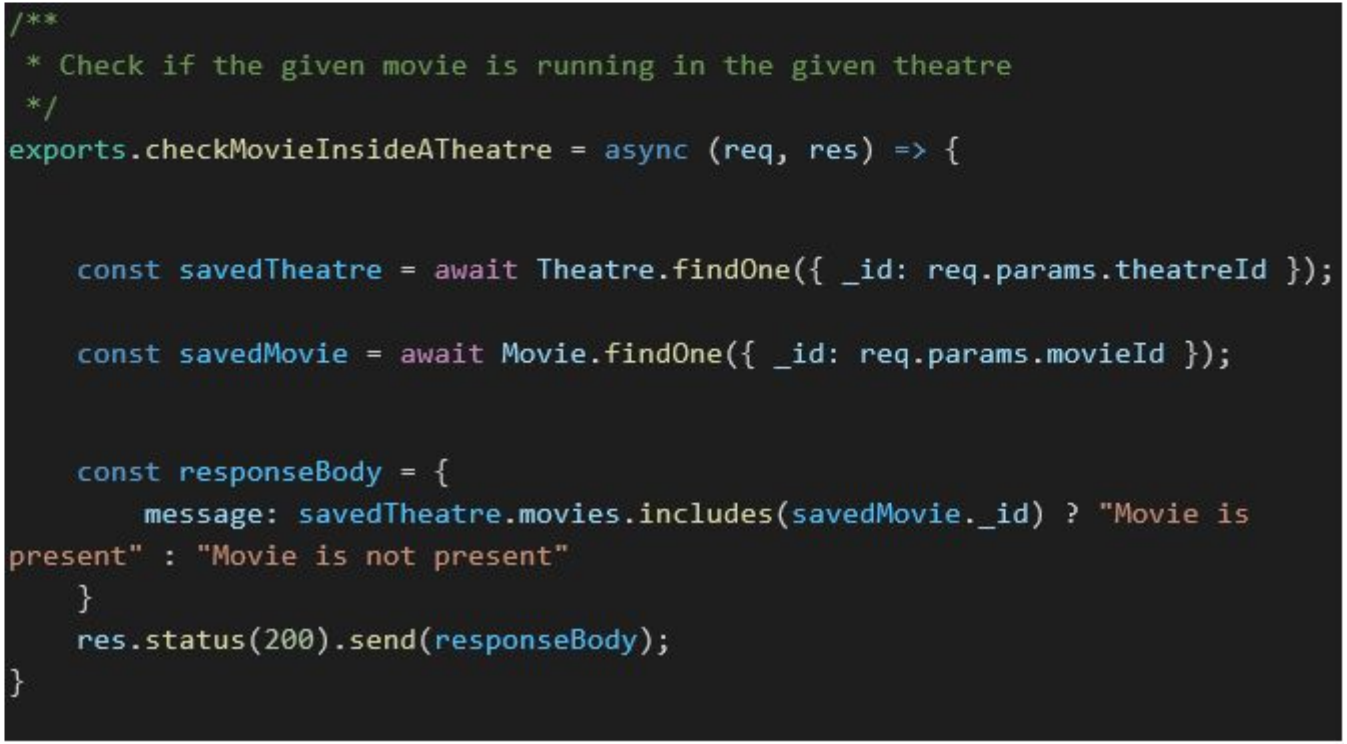
  GET /mba/api/v1/theatres/:threatreId/movies/:moviedId

  - **Creating a new function in Controller:**

We will create a new function inside theatre.controller.js named **checkMovieInsideATheatre.** Inside this we will first find the Theatre and Movie models from the given ids in the request params using findOne method and then will check if the movie id is present inside the movies array of the theatre model and send the response accordingly.

  - **Code Link:**

https://github.com/Vishwa07dev/mba_backend/blob/session3/controllers/theatre.controller.js



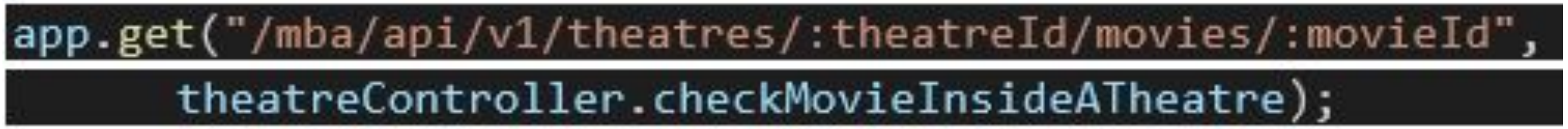
```
/**
 * Check if the given movie is running in the given theatre
 */
exports.checkMovieInsideATheatre = async (req, res) => {


    const savedTheatre = await Theatre.findOne({ _id: req.params.theatreId });

    const savedMovie = await Movie.findOne({ _id: req.params.movieId });


    const responseBody = {
        message: savedTheatre.movies.includes(savedMovie._id) ? "Movie is
present" : "Movie is not present"
    }
    res.status(200).send(responseBody);
}
```

- **Creating new route:**

We are using GET route to check whether the provided movie is running in the provided theatre.

- **Code Link:**

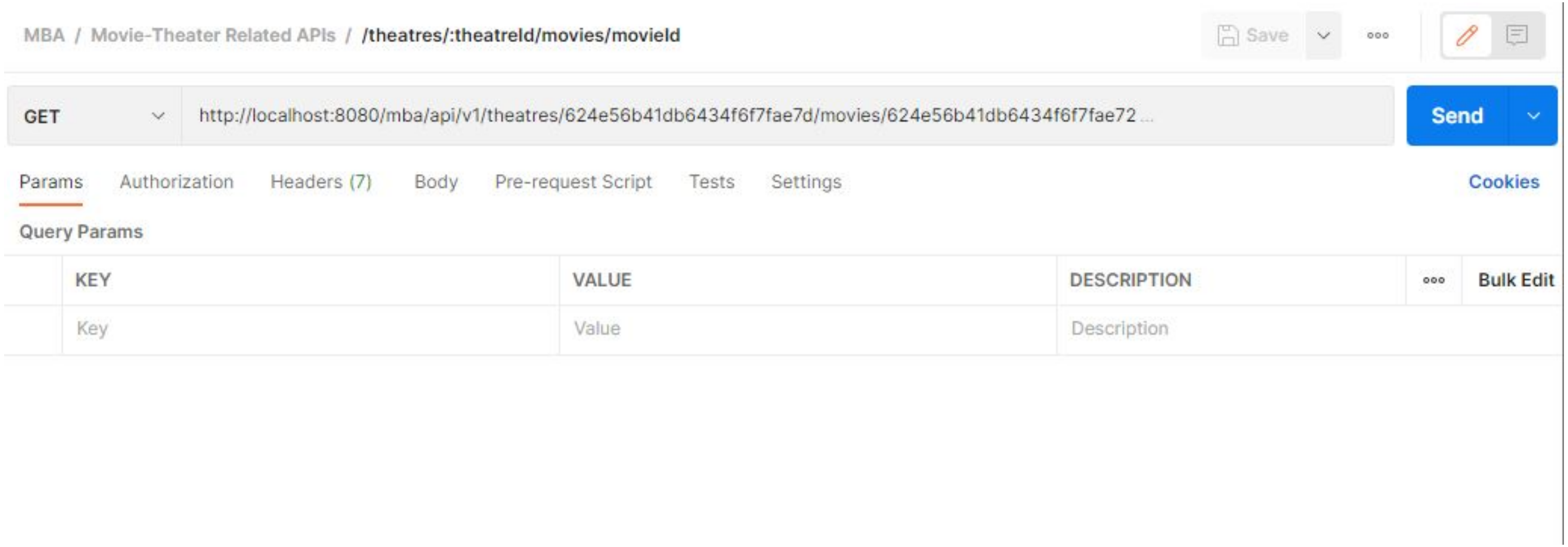
https://github.com/Vishwa07dev/mba_backend/blob/session3/routes/theatre.routes.js



```
app.get("/mba/api/v1/theatres/:theatreId/movies/:movieId",
      theatreController.checkMovieInsideATheatre);
```

Testing the API:
When a given movie is present inside the theatre:

Request:

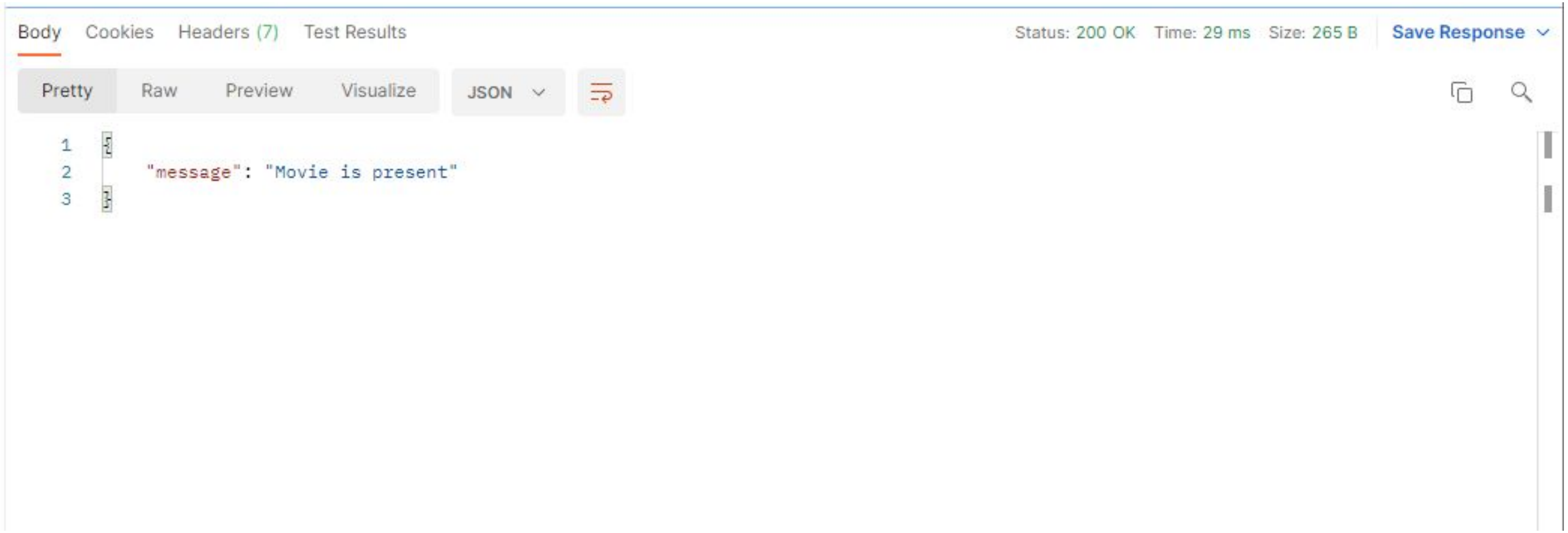


MBA / Movie-Theater Related APIs / /theatres/:theatreId/movies/movieId

GET http://localhost:8080/mba/api/v1/theatres/624e56b41db6434f6f7fae7d/movies/624e56b41db6434f6f7fae72 ...

KEY	VALUE	DESCRIPTION
Key	Value	Description

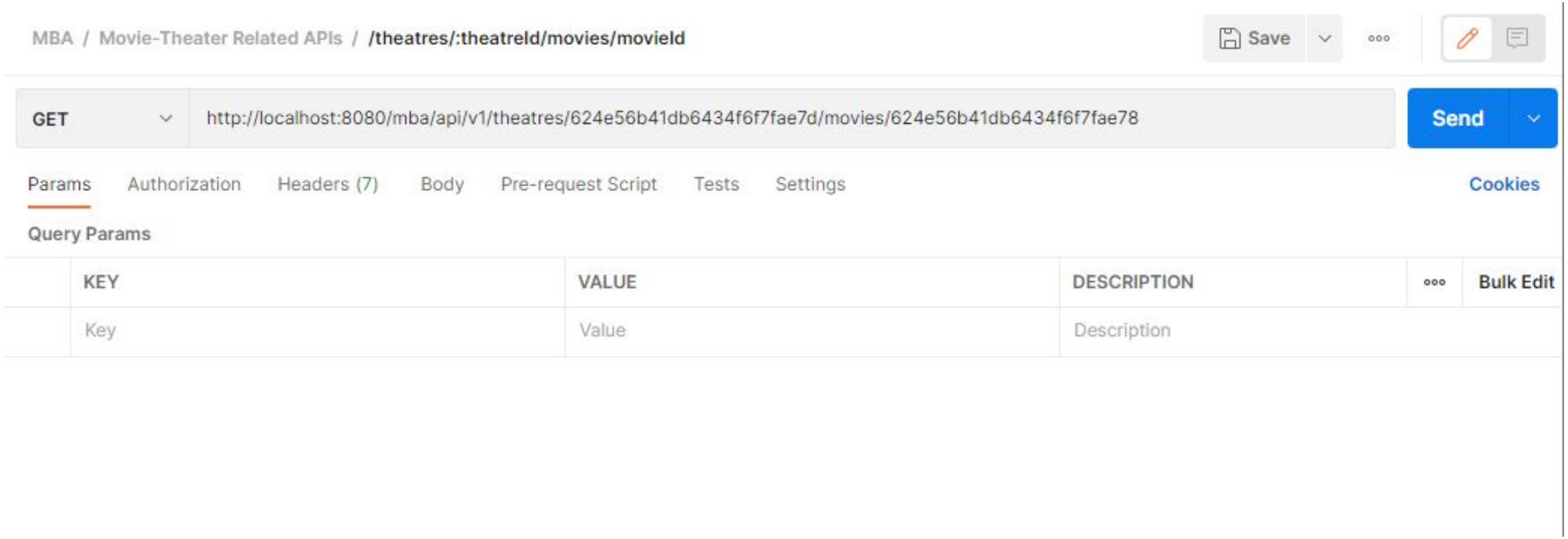
Response:



```
{
    "message": "Movie is present"
}
```

When a given movie is not present inside the theatre:

Request:

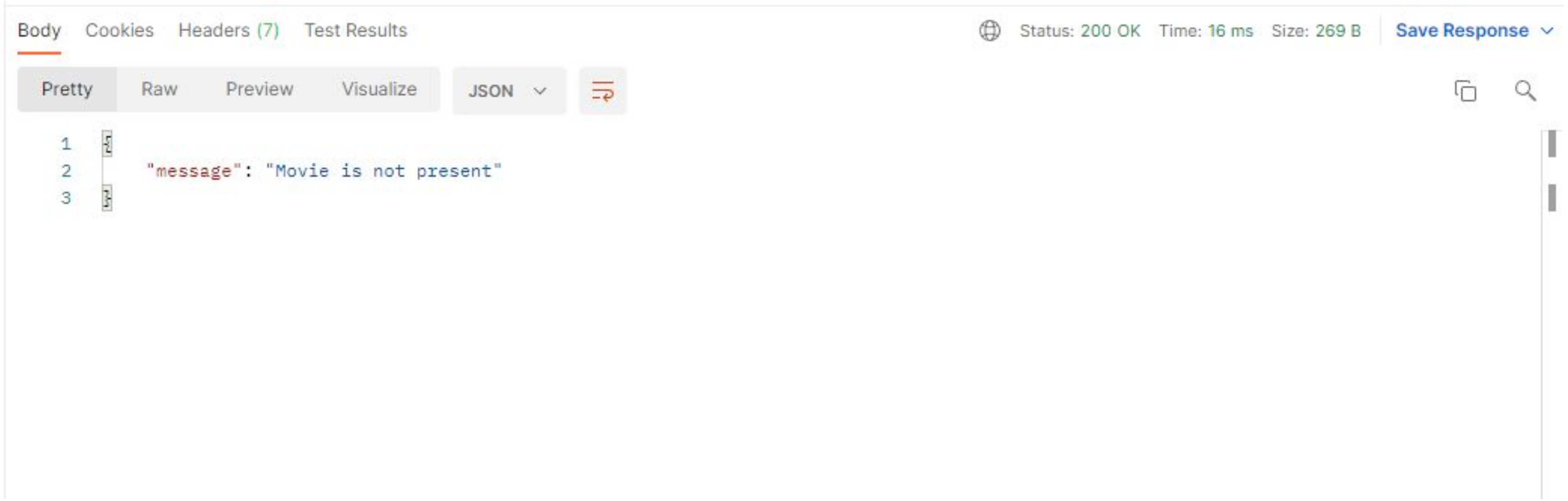


MBA / Movie-Theater Related APIs / /theatres/:theatreId/movies/movieId

GET http://localhost:8080/mba/api/v1/theatres/624e56b41db6434f6f7fae7d/movies/624e56b41db6434f6f7fae78

KEY	VALUE	DESCRIPTION
Key	Value	Description

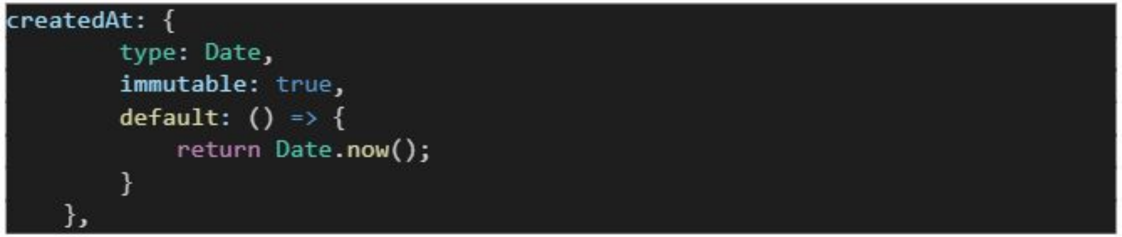
Response:



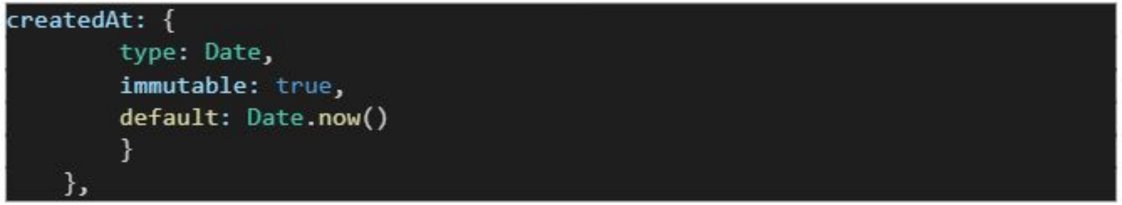
```
{
    "message": "Movie is not present"
}
```

MCQ Question 5 explained:

Explanation:



```
createdAt: {
        type: Date,
        immutable: true,
        default: () => {
            return Date.now();
        }
    },
```



```
createdAt: {
        type: Date,
        immutable: true,
        default: Date.now()
        }
    },
```

First one is used to assign the date when a new model object is created while the second one updates the date when the schema is defined.

Assignment questions

1. Create a middleware to verify if the given theatre id inside the request params is valid or if the theatre exist and return the response accordingly.
2. Test the middleware by passing an invalid theatre id first and another theatre id that do not exist.

In the upcoming class:

- Set up data model for user
- User registration
- Implementation and validation of JWT token
- Login API
- Update password
- Registration of system admin and client

Thank you