

# NODE JS RESTful API for Movie tracking system

## Introduction

This is a basic Node JS RESTful API for a simple movie tracking system. We manage movies and actors. We are using Postman for checking our routes. The movie schema has the following attributes

- i) Name (String)
- ii) Genre (String)
- iii) Actors (Array of Actor IDs, ObjectID associated with schema)
- iv) Business\_done (Number)
- v) Reviews (Array of Strings)
- vi) Rating (Number)

The Actor schema has the following attributes.

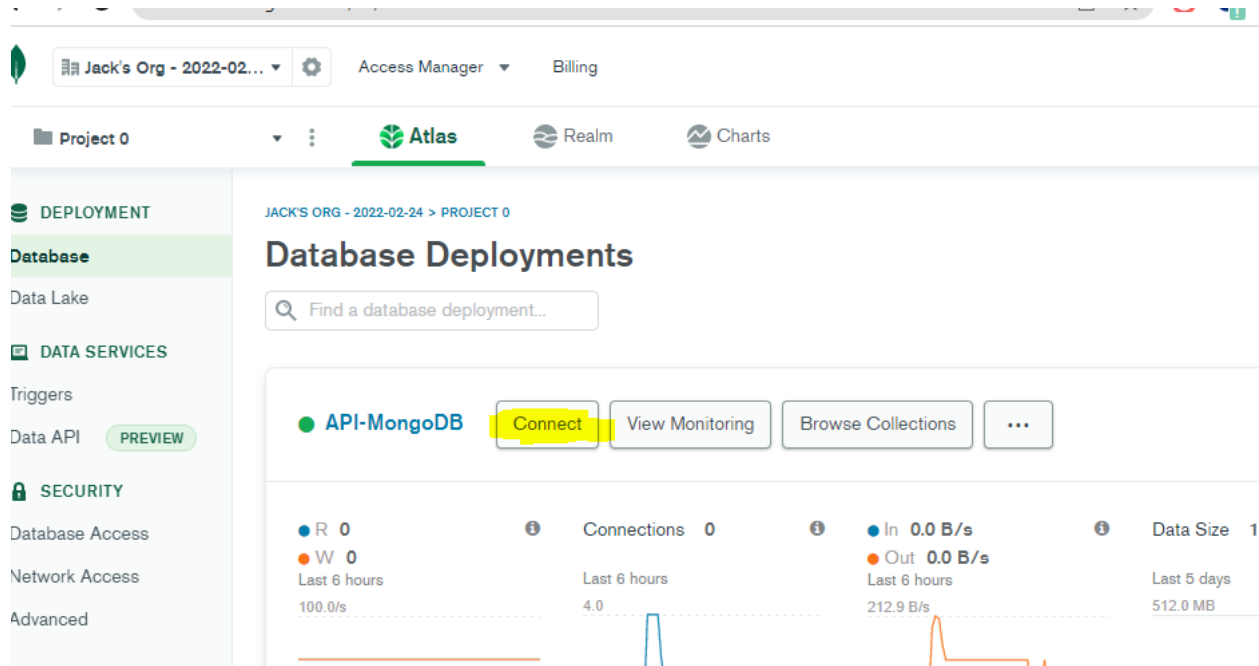
- i) Name (String)
- ii) Age (Number)
- iii) Gender (String)

## Pre Requisites

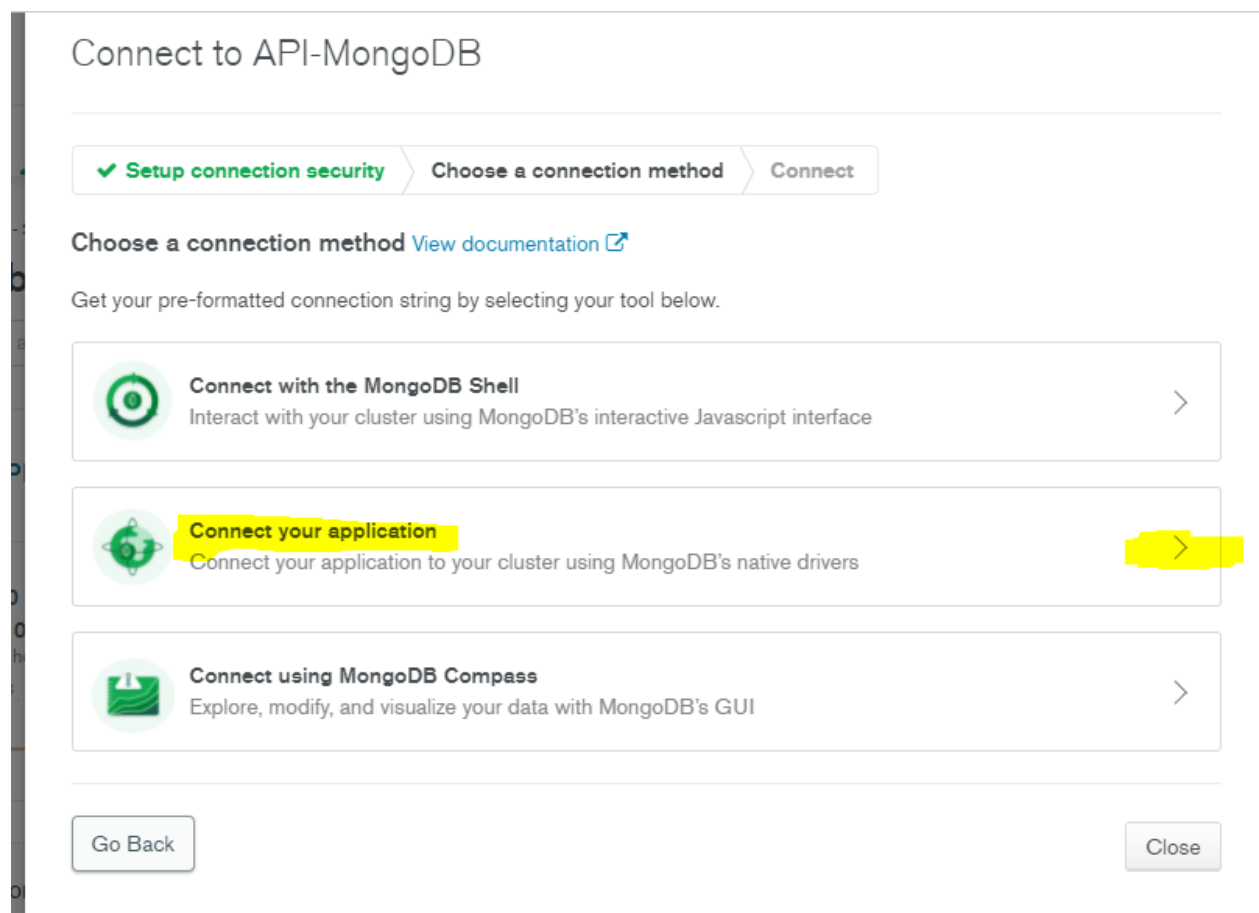
- i) Before we run our program, you need to sign up for mongoDB atlas. Go to the following link and sign up using yours google account.

<https://www.mongodb.com/cloud/atlas/register>

- ii) Now, sign in to the mongoDB atlas. You can see a connect button on the top left side. Simply click it.



iii) After this select the option “Connect your application” and then copy the URL.



✓ Setup connection security

✓ Choose a connection method

Connect

1

Select your driver and version

DRIVER

Node.js

VERSION

4.0 or later

2

Add your connection string into your application code

☐ Include full driver code example

mongodb+srv://<username>:<password>@api-mongodb.88z8c.mongodb.net/myFirstDatabase?  
retryWrites=true&w=majority

Replace **<password>** with the password for the **<username>** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back

Close

- iv) Now go to the app.js file in the project directory and paste the URL in line 11. Enclose it in double quotes as well. In the URL you can see <password>, simply type your mongoDB atlas account password here.

```
app.js 2 ...
1 const bodyParser = require('body-parser');
2
3 const mongoose = require("mongoose");
4 const formidable = require('express-formidable');
5 const app = express();
6
7 const actorRoutes = require('./api/routes/actors');
8 const movieRoutes = require('./api/routes/movies');
9 const userRoutes = require('./api/routes/users');
10
11 | mongoose.connect('');
12
13 app.use(morgan('dev'));
14 app.use(bodyParser.urlencoded({extended: false}));
15 //app.use(formidable());
```

Paste URL

- v) After following the above steps run the following command in the command prompt and your program should run

Command: npm start

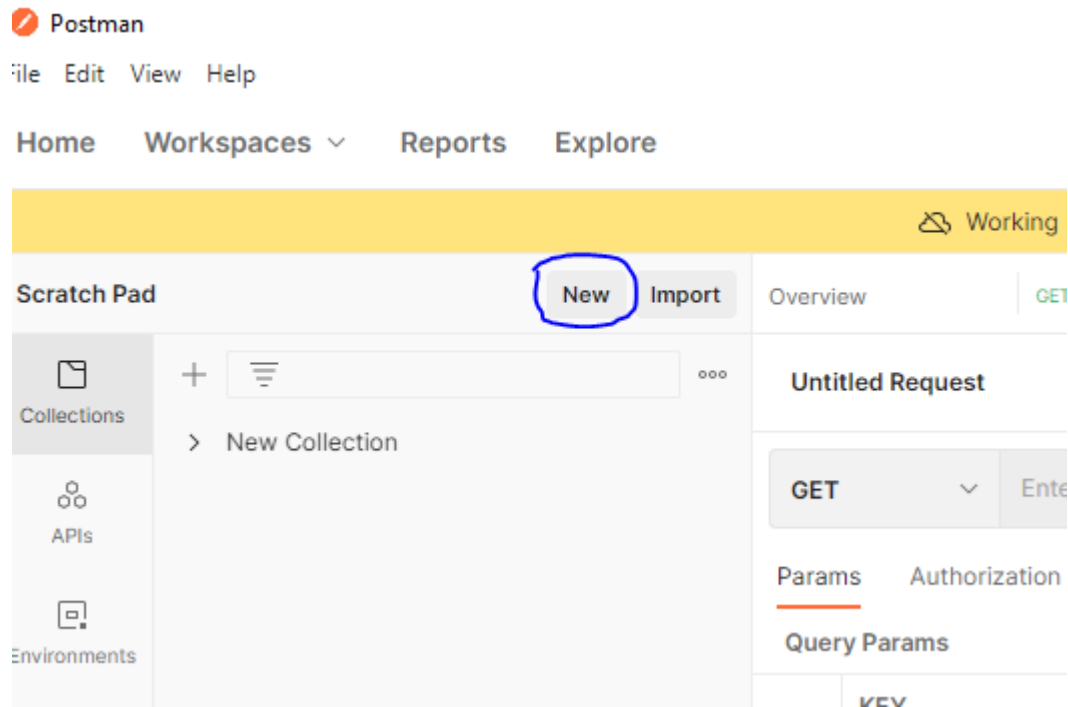
## Installing Postman

Now, we are going to check and test our routes. We would need to install Postman for this.

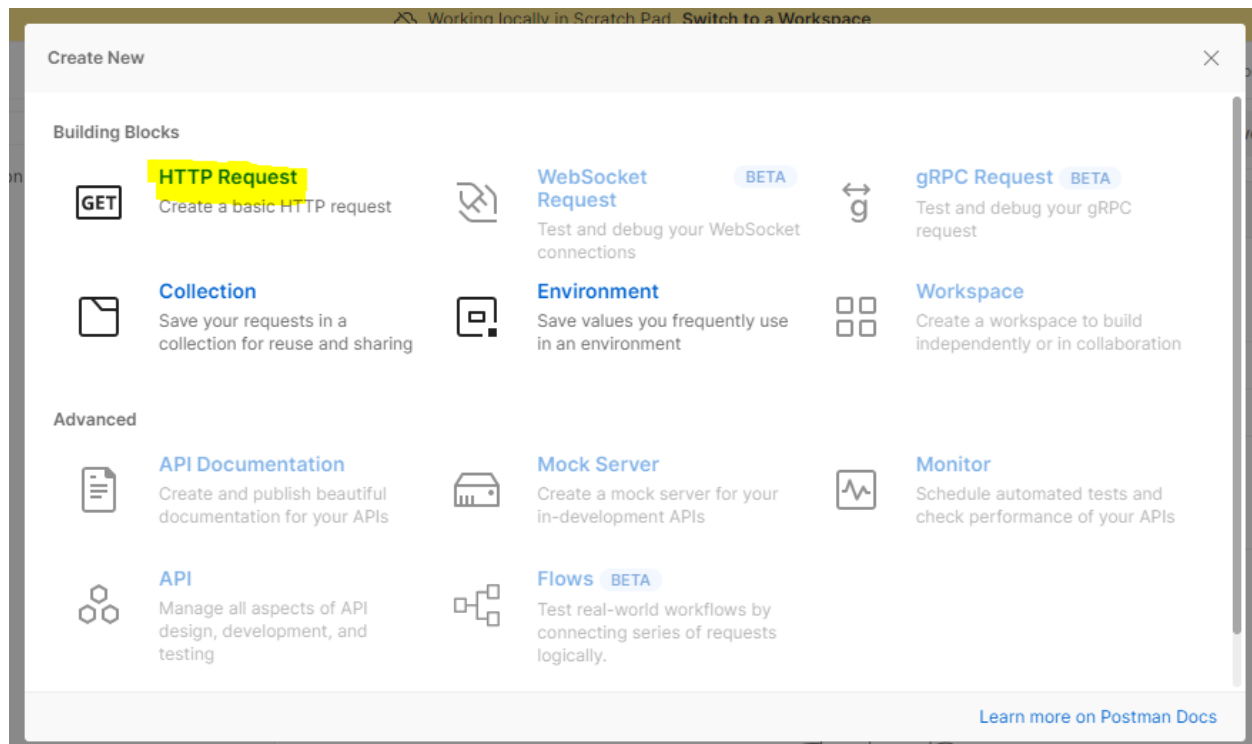
- i) Go to the following link and download Postman.

<https://www.postman.com/>

- ii) After this, run the installer and install it.
- iii) After this is done, start the Postman.
- iv) Select New in the top left column



- v) Select HTTP request.



## Signup Route

Now, go to the following URL and sign up a new user.

Localhost:3000/users/signup

Refer to the following picture for the data.

The screenshot shows a REST client interface with the following details:

- Request:** Method **POST**, URL **localhost:3000/users/signup**. The request body is a JSON object: 

```
{  "name": "Jack",  "email": "jack@gmail.com",  "phoneNo": "03609191910",  "password": "jack123"}
```
- Response:** Status **201 Created**, Time **1058 ms**, Size **383 B**. The response body is a JSON object: 

```
{  "message": "User Created"}
```
- Interface Elements:** Tabs for Params, Auth, Headers (8), Body (selected), Pre-req., Tests, and Settings. A 'Send' button is present. The response is displayed in 'Pretty' format.

## Login Route

Now, go the following route and enter the login data.

Localhost:3000/users/login

The screenshot shows a REST client interface with the following details:

- URL:** localhost:3000/users/login
- Method:** POST
- Request Body (JSON):**

```
{  "email": "jack@gmail.com",  "password": "jack123"}
```
- Response Status:** 200 OK, 754 ms, 599 B
- Response Body (JSON):**

```
{  "message": "Auth Successfull!",  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFnZWVudCI6ImphY2tAZ21haWwudY29tIiwiaWQiOiI2MjFkZjcwN2U4ZDF1NzRhbnJkbnUzYWMiLCJpYXQiOiE2NDYxMzExNDMsImV4cCI6MTY0NjEzNDc0M30. GFRRpTrJ06xfkzX7wfre6t1KE-nbPvvnMF7M7aFKxJJA"}
```

After, we logged in a token is generated. We need to pass this token to header so that we can authorize the user. Simply copy the token and then go to the header tab and create a key named Authorization. In the value field, type Bearer and after a single space paste the token here.

Working locally in Scratch Pad. Switch to a Workspace

OverviewPOST localhost:3000/users/loginErrorPOST localhost:3000/users/login+...No Environment

localhost:3000/users/loginSave


POSTlocalhost:3000/users/loginSend

ParamsAuthHeaders (10)BodyPre-reqTestsSettingsCookies

Headers8 hidden

	KEY	VALUE	DES		Bulk Edit	Presets
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5c...				
<input checked="" type="checkbox"/>	Content-Type	application/json				
	Key	Value	Description			

Response



Could not send request

Create another key named Content-Type and enter the text “application/json” in the value field.

## Get All Actor Route

Now we can check and test different routes for actor and movie schema. Following route will give us all the actors.



The screenshot shows a REST client interface with the following details:

- URL:** localhost:3000/actors
- Method:** GET
- Headers:** 8 hidden. Visible headers include:

KEY	VALUE	DES	...	Bulk Edit	Presets
Authorization	Bearer eyJhbGciOiJIUzI1NiIsInR5c...				
Content-Type	application/json				
Key	Value	Description			
- Body:** Pretty, Raw, Preview, Visualize. JSON format selected.

```
1 {
2   "count": 5,
3   "actor": [
4     {
5       "_id": "62176a85604e6b148fc7da56",
6       "name": "Salman Khan",
7     }
8   ]
9 }
```
- Status:** 201 Created, 468 ms, 774 B
- Buttons:** Save, Send, Cookies, Save Response, Runner, Trash

## Post A New Actor Route

In the following route, we can add new actor to the database.

Working locally in Scratch Pad. Switch to a Workspace

ort Overview × POST localhost: Error POST localhost: + ... No Environment

localhost:3000/actors Save

POST localhost:3000/actors Send

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1 {
2   "name": "John",
3   "age": "32",
4   "gender": "Male"
5 }
```

Body 201 Created 308 ms 536 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Actor created successfully!",
3   "actor": {
4     "name": "John",
5     "age": 32,
6     "gender": "Male",
7   }
8 }
```

Runner Trash

## Get an Individual Actor Route

In the following route, we can get an individual actor by giving an id of the actor. We can get id of the actor by executing the following route.

**Route type:** get

**Route URL:** localhost:3000/actors

port Overview GET localhost ● Error POST localho: ● + ... No Environment

... localhost:3000/actors/621df9c1e8d1e74a690653b0 Save

GET localhost:3000/actors/621df9c1e8d1e74a690653b0 Send

Params Auth Headers (10) Body ● Pre-req. Tests Settings

Query Params

KEY	VALUE	DESCRIPTION	...	E
Key	Value	Description		

Body 200 OK 353 ms 499 B Save Respo

Pretty Raw Preview Visualize JSON

```

1 {
2   "actor": {
3     "_id": "621df9c1e8d1e74a690653b0",
4     "name": "John",
5     "age": 32,
6     "gender": "Male"
  }
}
```

Runner

## Patch/Update an Actor Route

In the following route we can update an actor by giving an id of the actor. We can get id of the actor by executing the following route.

**Route type:** get

**Route URL:** localhost:3000/actors

Working locally in Scratch Pad. Switch to a Workspace

port Overview PATCH localhc Error POST localho: + ... No Environment

localhost:3000/actors/621df9c1e8d1e74a690653b0 Save

PATCH localhost:3000/actors/621df9c1e8d1e74a690653b0 Send

Params Auth Headers (10) Body Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1 [
2   {
3     "propName": "age", "value": "35"
4   }
5 ]
```

Body 200 OK 297 ms 470 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Actor Updated!",
3   "request": {
4     "type": "PATCH",
5     "url": "http://localhost:3000/actors/621df9c1e8d1e74a690653b0"
6   }
7 }
```

Runner Traces

In the propName, we need to provide the name of the column which we want to update and in the value column we need to provide the new value.

## Get All Movies Route

Now, we are going to test the movie route. We can get all the movies by giving the following route.

Working locally in Scratch Pad. Switch to a Workspace

OverviewGET localhostErrorPOST localho: +...No Environment

localhost:3000/movies/Save

GETlocalhost:3000/movies/Send

ParamsAuthHeaders (10)BodyPre-req.TestsSettingsCookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body201 Created747 ms799 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "count": 2,
3   "movie": [
4     {
5       "name": "Sanam",
6       "genre": "love story",
7       ...
8     }
9   ]
10 }
```

## Post a New Movie Route

We can add a new movie by providing the data given in the picture below. In the ActorID column, we need to give the ids of the actors that are already registered with us otherwise this will give an error.

re

Search Postman

Working locally in Scratch Pad. Switch to a Workspace

Import Overview POST localhost: Error POST localhost: + ... No Environment

localhost:3000/movies/ Save

POST localhost:3000/movies/ Send

Params Auth Headers (10) Body Pre-req. Tests Settings

raw JSON Beautify

```
1 {
2   "name": "Super Man",
3   "genre": "action",
4   "actorID": [
5     "62176a85604e6b148fc7da56",
6     "621df9c1e8d1e74a690653b0"
7   ],
8   "business_done": "12000000"
9 }
```

Body 201 Created 718 ms 637 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Movie created successfully!",
3   "actor": {
```

Runner Trash

## Get an Individual Movie Route

We can get an individual movie by passing its id to the following route. We can get id of the movie by getting all the movies.

**Route type:** get

**Route URL:** localhost:3000/movies

Working locally in Scratch Pad. Switch to a Workspace

port

OverviewGET localhostErrorPOST localho+

No Environment

localhost:3000/movies/621dfbd4e8d1e74a690653b9

Save

GET

localhost:3000/movies/621dfbd4e8d1e74a690653b9

Send

ParamsAuthHeaders (10)BodyPre-reqTestsSettingsCookies

Query Params

	KEY	VALUE	DESCRIPTION		Bulk Edit
	Key	Value	Description		

Body

201 Created858 ms722 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "movie": {
3     "_id": "621dfbd4e8d1e74a690653b9",
4     "name": "Super Man",
5     "genre": "action",
6     "actors": [
7       {
8         "_id": "62176a85604e6b148fc7da56",
9         "name": "Salman Khan",
10        "age": 53,
11      }
12    ]
13  }
14 }
```

RunnerTrash

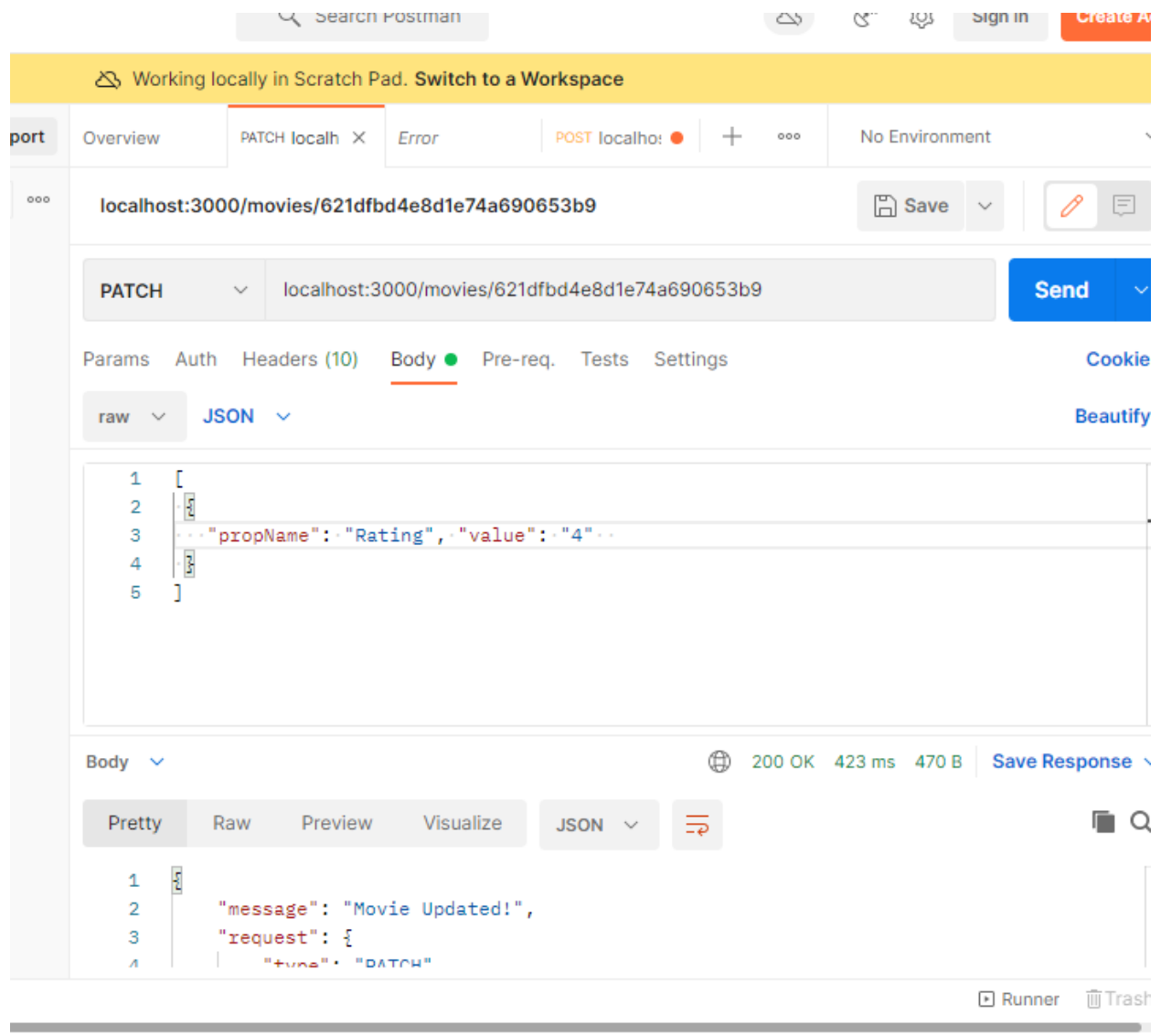
## Patch/Update a Movie Route

We can update a movie by giving its id to the route. We can give multiple reviews to a movie. Also the Rating should be from 1 to 5.

We can get id of the movie by executing the following route.

**Route type:** get

**Route URL:** localhost:3000/movies



## Delete a Movie Route

At the last we can delete a movie by providing an id to the following route. We can get id of the movie by executing the following route.

**Route type:** get

**Route URL:** localhost:3000/movies



Working locally in Scratch Pad. Switch to a Workspace

Overview

DEL localhost

Error

POST localho

+

...

No Environment

localhost:3000/movies/621dfbd4e8d1e74a690653b9

Save

Send

DELETE

localhost:3000/movies/621dfbd4e8d1e74a690653b9

Send

ParamsAuthHeaders (10)BodyPre-req.TestsSettingsCookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body

200 OK370 ms530 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "message": "Movie Deleted!",
3   "request": {
4     "type": "POST",
5     "url": "http://localhost:3000/movies/",
6     "data": {
7       "name": "String",
8       "genre": "String",
9       "actors": "String",
```

RunnerTrash