Integrate Authentication in Theatre APIs

Relevel
by Unacademy

When we use application we are the customers while the application owners act as an administrator with many authorities and access to private routes of the application. In this class we will learn how to distinguish the accessibility of different endpoints according to user access and privileges.

Class Agenda

- So far we have implemented authentication and authorization in our application.
- Now it’s time to see those middlewares like verifyToken and isAdmin in action.
- We will be dividing route into two parts one which will be accessible to logged in user i.e., those who has a valid token
- And the rest accessible only to admins also with a valid token.

From the previous class:

- So far, in our Movie Booking Application we have created our CRUD end points on Movie and Theatre by creating corresponding models, controller and routes using mongoDB and express and tested the same using Postman.
- Then we Created User model for storing customer, clients and admin details and create endpoints for login, register and so on.
- We have also created some middleware to add validation to post and put request of theatres and movies.
- In this class we will be using that middleware to protect our endpoints we have created so far and understand different validations for different types of routes.

List of Concepts Involved

In this class, we will learn the following items:

- Add authentication in theatre APIs
- Add authentication and authorization on admin APIs-
    1. Admins can Create/Update/Delete any movies.
    2. Admins can Create/Update/Delete any movies in any theatre
    3. Admins can Create/Update/Delete any theatres
    4. Admin can update the details of any type of users.
- Only authenticated user should be allowed to use any other API

Add authentication in theatre APIs

- We will begin by learning how we can add authentication in an API.
- We need to make all the theatre APIs accessible only to logged in users.
- We have already created a signin route which gives us a token on successful response.
- All we need is to validate this token by having middleware function to do so.
- We will first go through the middleware and learn how we can use it to protect routes for theatre resource.
- Let us begin step by step.

Add authentication in theatre APIs

Step 1: Middleware to Use:

Code link:

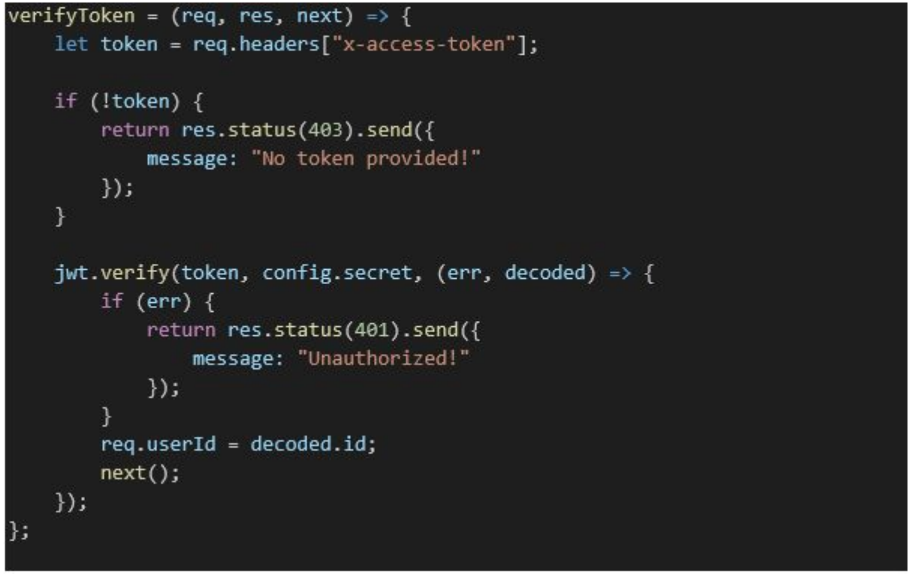
https://github.com/Vishwa07dev/mba_backend/blob/session6/middlewares/authjwt.js

- verifyToken middleware first extract the token which is expected to be sent in request header with a token name of “x-access-token” , a unique name to avoid overlap with any other headers.
- If the token is not present it sends a response of “no token provided” with a forbidden status code of 403.
- If token I present it verifies the token using jwt verify function which if result with an error then sends a response of Unauthorized and status code of 401.
- If the token is valid we append the request body with user id stored in the token and proceed with next function which is later used in the controller for verification of user.

Add authentication in theatre APIs

- Here, as we can see once the jwt token is verified i.e. no error is thrown we have added userId in decoded object as payload.
- This userId acts as an identifier which is used to verify the token associated with the specific user using it and not any other user trying to copy the token and access the apis incorrectly.

Add authentication in theatre APIs



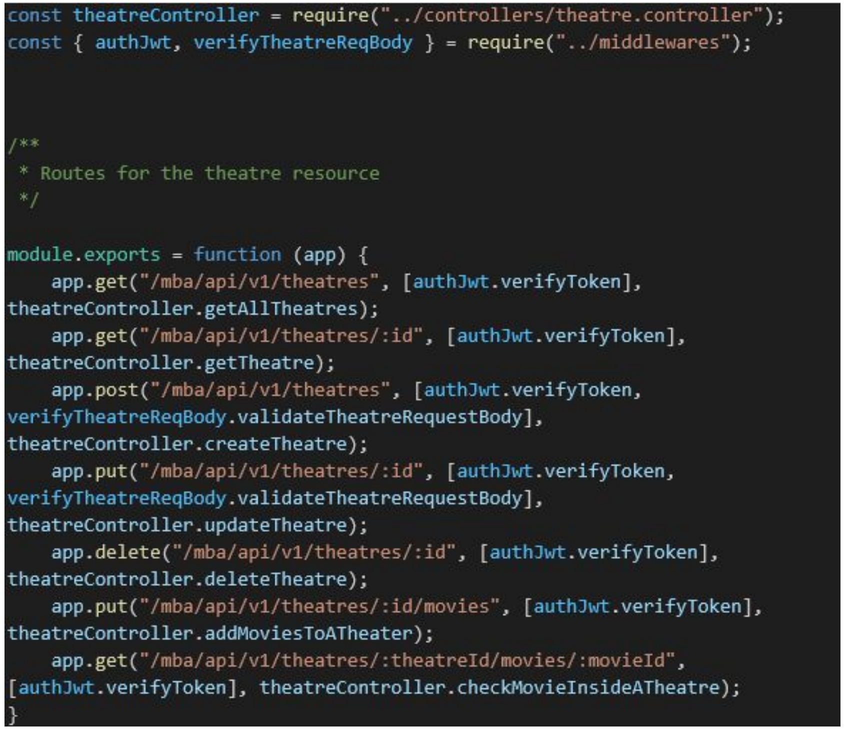
```
verifyToken = (req, res, next) => {
    let token = req.headers["x-access-token"];

    if (!token) {
        return res.status(403).send({
            message: "No token provided!"
        });
    }

    jwt.verify(token, config.secret, (err, decoded) => {
        if (err) {
            return res.status(401).send({
                message: "Unauthorized!"
            });
        }
        req.userId = decoded.id;
        next();
    });
};
```

Add authentication in theatre APIs

Step 2: Adding middleware in the routes:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/theatre.routes.js

- As you can see in from the code in each route we have added a function in the middle authJwt.verifyToken which as explained earlier validates that the user accessing these routes have a valid token.

Add authentication in theatre APIs



```
const theatreController = require("../controllers/theatre.controller");
const { authJwt, verifyTheatreReqBody } = require("../middlewares");



/**
 * Routes for the theatre resource
 */

module.exports = function (app) {
    app.get("/mba/api/v1/theatres", [authJwt.verifyToken],
theatreController.getAllTheatres);
    app.get("/mba/api/v1/theatres/:id", [authJwt.verifyToken],
theatreController.getTheatre);
    app.post("/mba/api/v1/theatres", [authJwt.verifyToken,
verifyTheatreReqBody.validateTheatreRequestBody],
theatreController.createTheatre);
    app.put("/mba/api/v1/theatres/:id", [authJwt.verifyToken,
verifyTheatreReqBody.validateTheatreRequestBody],
theatreController.updateTheatre);
    app.delete("/mba/api/v1/theatres/:id", [authJwt.verifyToken],
theatreController.deleteTheatre);
    app.put("/mba/api/v1/theatres/:id/movies", [authJwt.verifyToken],
theatreController.addMoviesToATheater);
    app.get("/mba/api/v1/theatres/:theatreId/movies/:movieId",
[authJwt.verifyToken], theatreController.checkMovieInsideATheatre);
}
```

Testing the APIs:

Let us now test the APIs by sending a request :

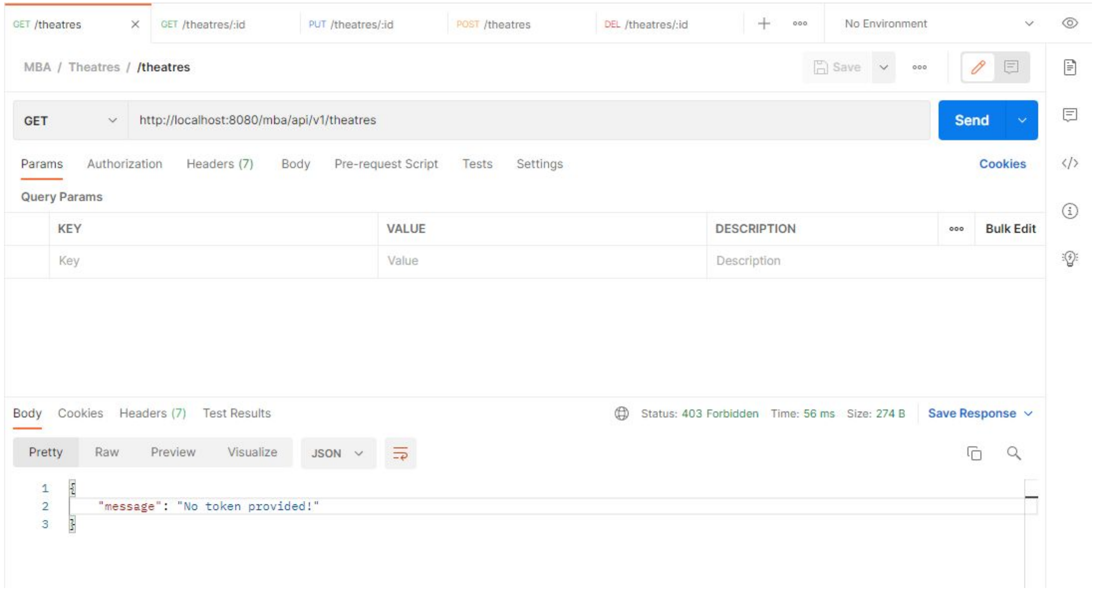
1. Without a token
2. With a token

And analyse the result.

GET /mba/api/v1/theatres

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



GET http://localhost:8080/mba/api/v1/theatres

KEY	VALUE	DESCRIPTION
Key	Value	Description

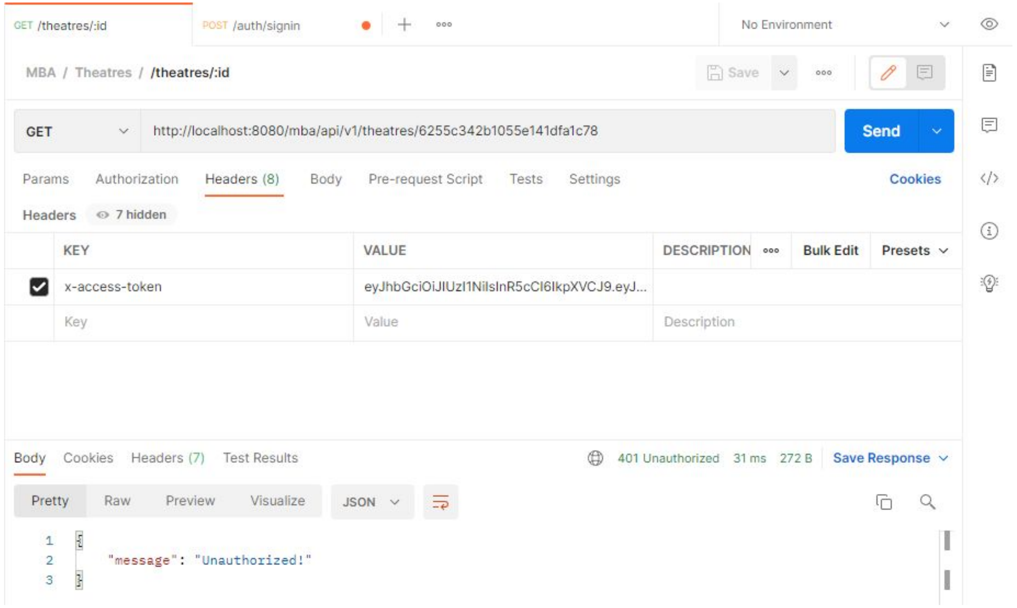
Status: 403 Forbidden Time: 56 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

GET /mba/api/v1/theatres

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is sent.



GET /theatres/:id   POST /auth/signin   No Environment

MBA / Theatres / /theatres/:id

GET http://localhost:8080/mba/api/v1/theatres/6255c342b1055e141dfa1c78

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ...	
	Key	Value	Description

Body Cookies Headers (7) Test Results   401 Unauthorized 31 ms 272 B Save Response

Pretty Raw Preview Visualize JSON

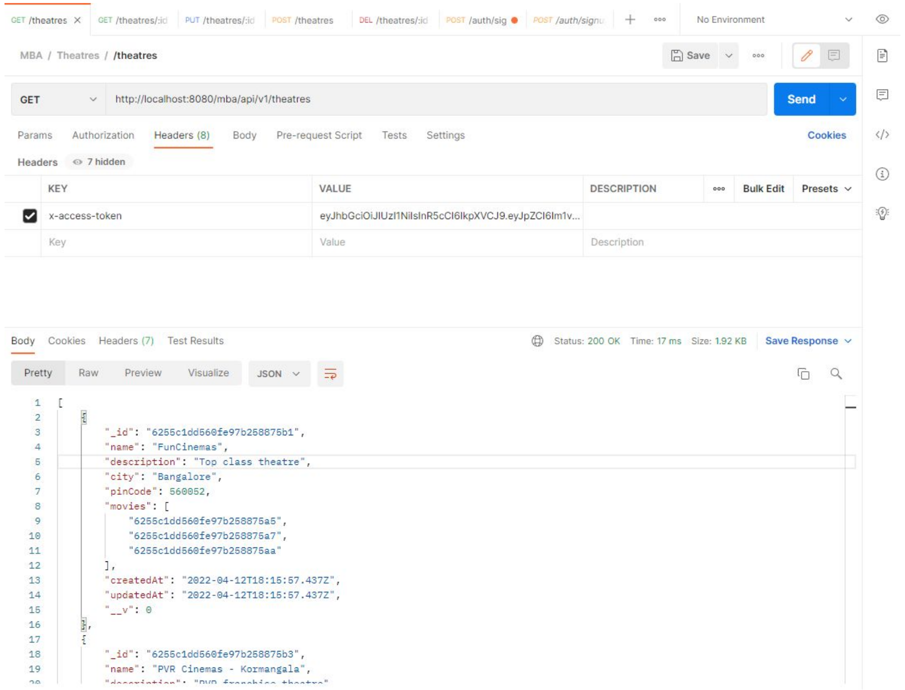
```
{
    "message": "Unauthorized!"
}
```

GET /mba/api/v1/theatres

With a token:

- Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.
- We have received a successful response here.
- Note: we can get the token by creating an account and signing in using signup and signin routes

GET /mba/api/v1/theatres



MBA / Theatres / /theatres

GET http://localhost:8080/mba/api/v1/theatres

Headers (8)

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

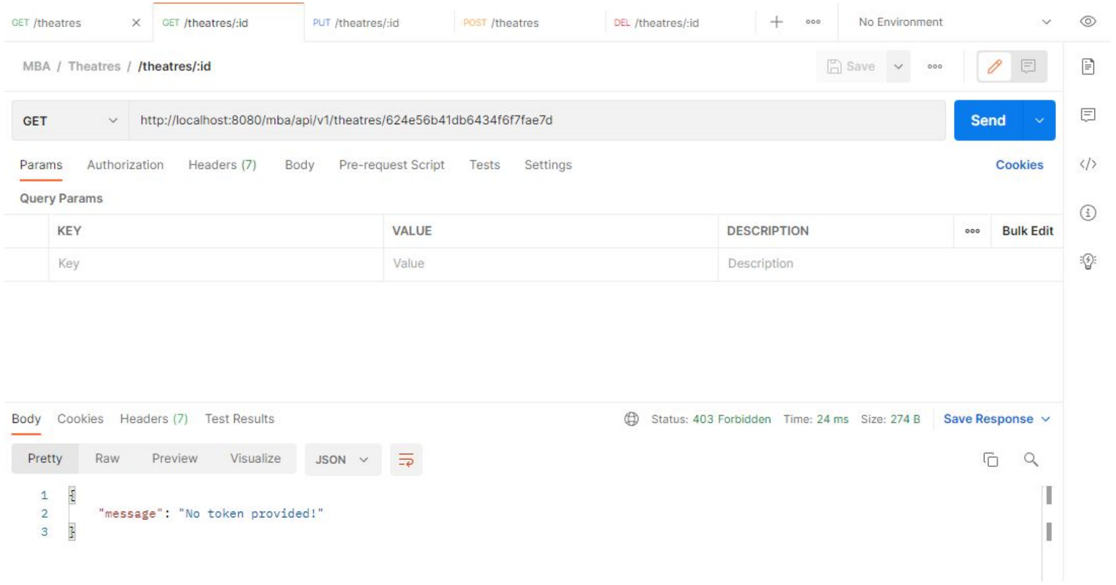
Status: 200 OK Time: 17 ms Size: 1.92 KB

```
[
    {
        "_id": "6255c1dd560fe97b258875b1",
        "name": "FunCinemas",
        "description": "Top class theatre",
        "city": "Bangalore",
        "pinCode": 560052,
        "movies": [
            "6255c1dd560fe97b258875a5",
            "6255c1dd560fe97b258875a7",
            "6255c1dd560fe97b258875aa"
        ],
        "createdAt": "2022-04-12T18:15:57.437Z",
        "updatedAt": "2022-04-12T18:15:57.437Z",
        "__v": 0
    },
    {
        "_id": "6255c1dd560fe97b258875b3",
        "name": "PVR Cinemas - Kormangala",
```

GET /mba/api/v1/theatres/:id

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



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

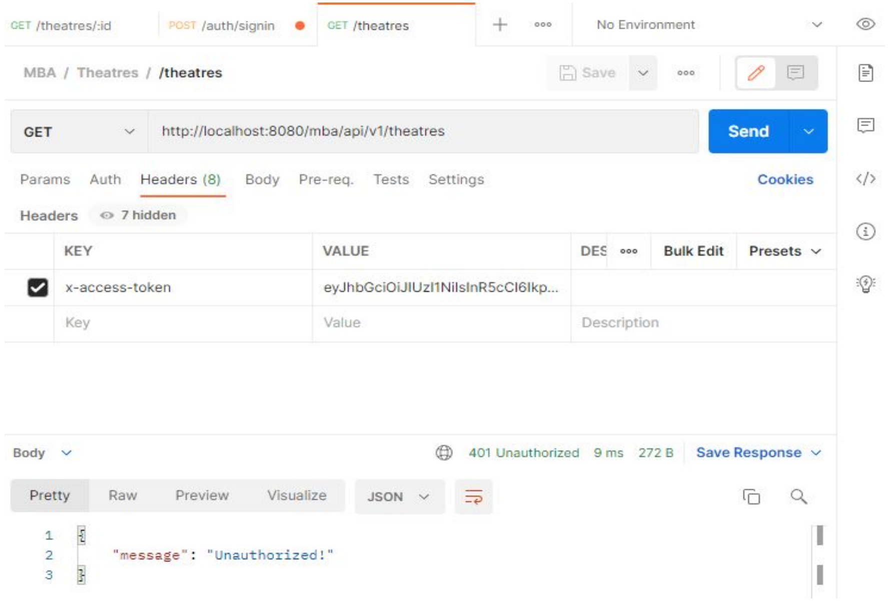
Status: 403 Forbidden Time: 24 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

GET /mba/api/v1/theatres/:id

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is sent.



GET http://localhost:8080/mba/api/v1/theatres

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp...	
Key	Value	Description

401 Unauthorized 9 ms 272 B

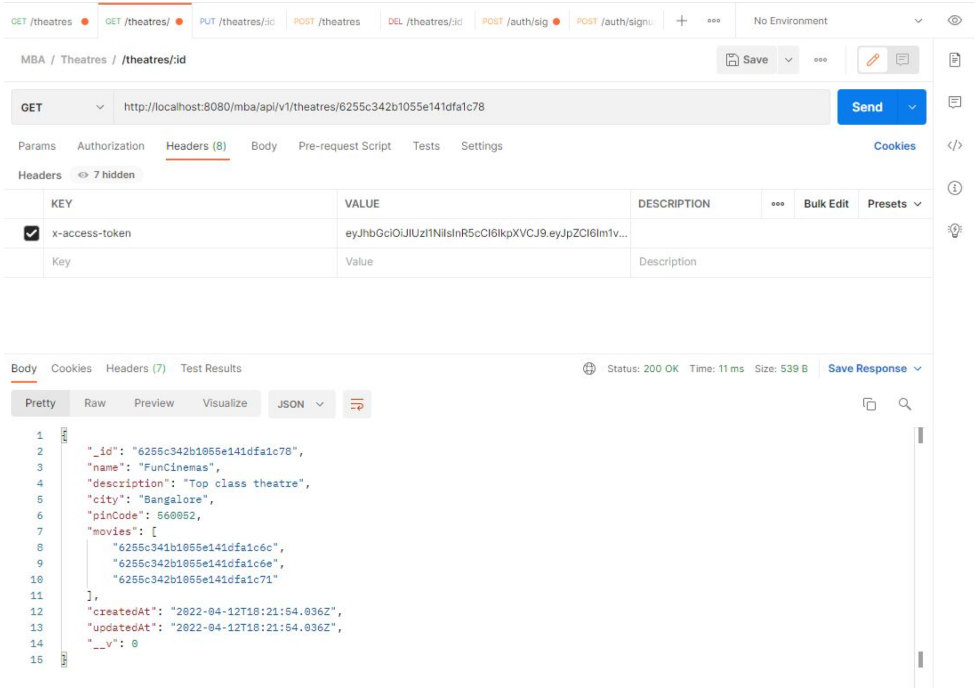
```
{
    "message": "Unauthorized!"
}
```

GET /mba/api/v1/theatres/:id

With a token:

- Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.
- We have received a successful response here.
- Note: we can get the token by creating an account and signing in using signup and signin routes

GET /mba/api/v1/theatres/:id



MBA / Theatres / /theatres/:id

GET http://localhost:8080/mba/api/v1/theatres/6255c342b1055e141dfa1c78

Headers (8)

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	

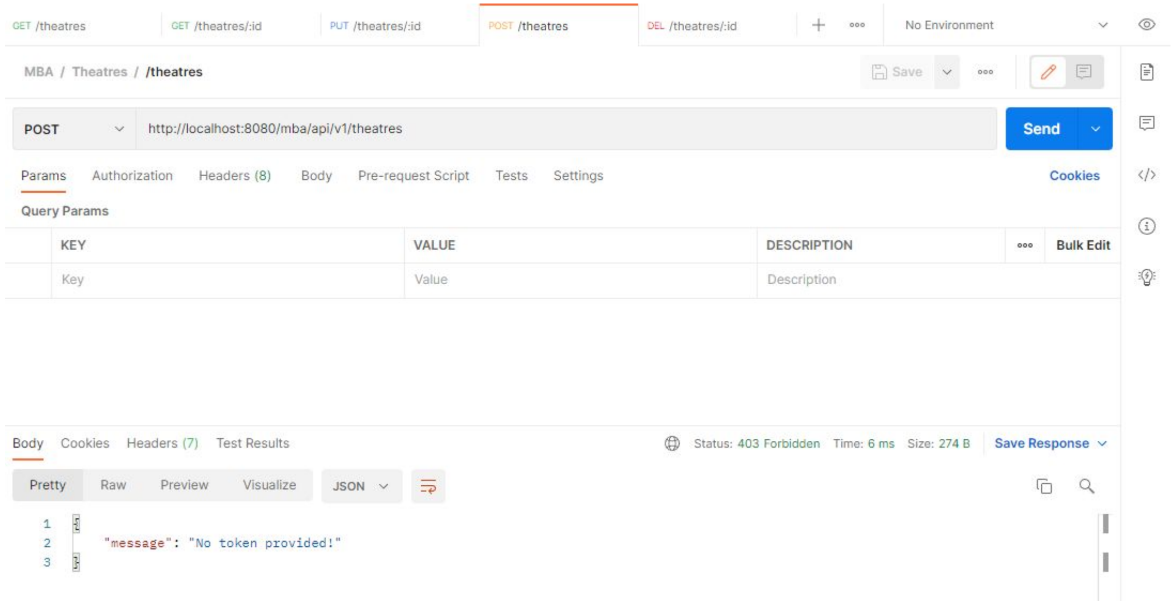
Status: 200 OK Time: 11 ms Size: 539 B

```
{
    "_id": "6255c342b1055e141dfa1c78",
    "name": "FunCinemas",
    "description": "Top class theatre",
    "city": "Bangalore",
    "pinCode": 560052,
    "movies": [
        "6255c341b1055e141dfa1c6c",
        "6255c342b1055e141dfa1c6e",
        "6255c342b1055e141dfa1c71"
    ],
    "createdAt": "2022-04-12T18:21:54.036Z",
    "updatedAt": "2022-04-12T18:21:54.036Z",
    "__v": 0
}
```

POST /mba/api/v1/theatres

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



POST http://localhost:8080/mba/api/v1/theatres

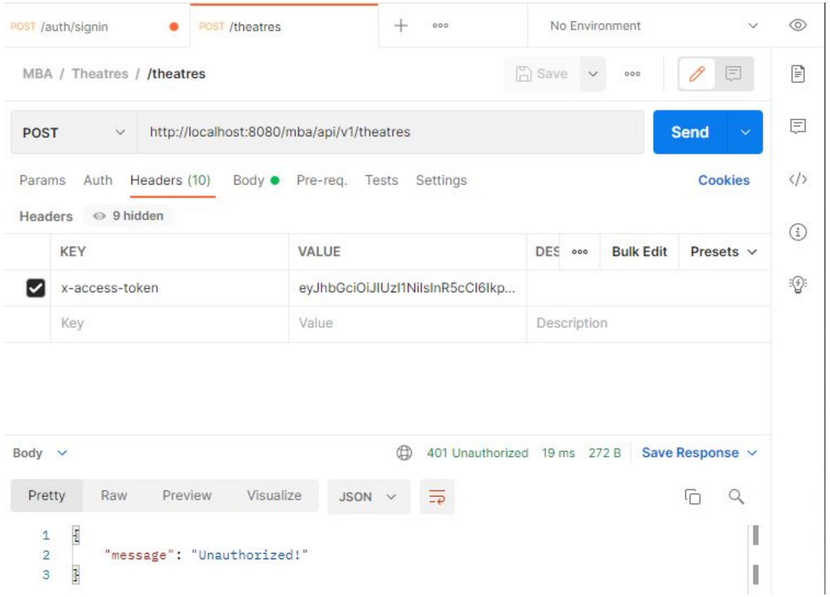
Status: 403 Forbidden Time: 6 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

POST /mba/api/v1/theatres

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is sent.



POST http://localhost:8080/mba/api/v1/theatres

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp...	
Key	Value	Description

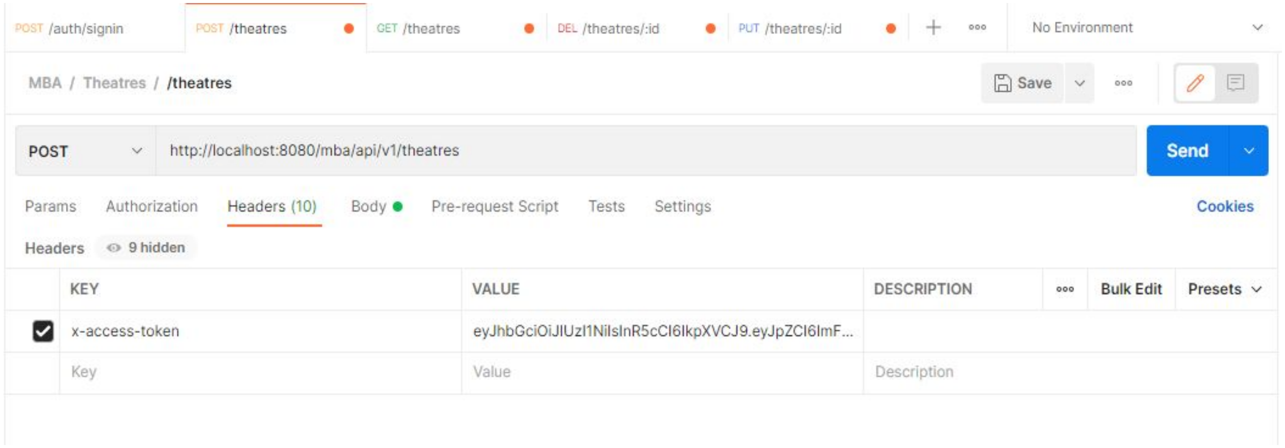
401 Unauthorized 19 ms 272 B

```
{
    "message": "Unauthorized!"
}
```

POST /mba/api/v1/theatres

With a token

Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.

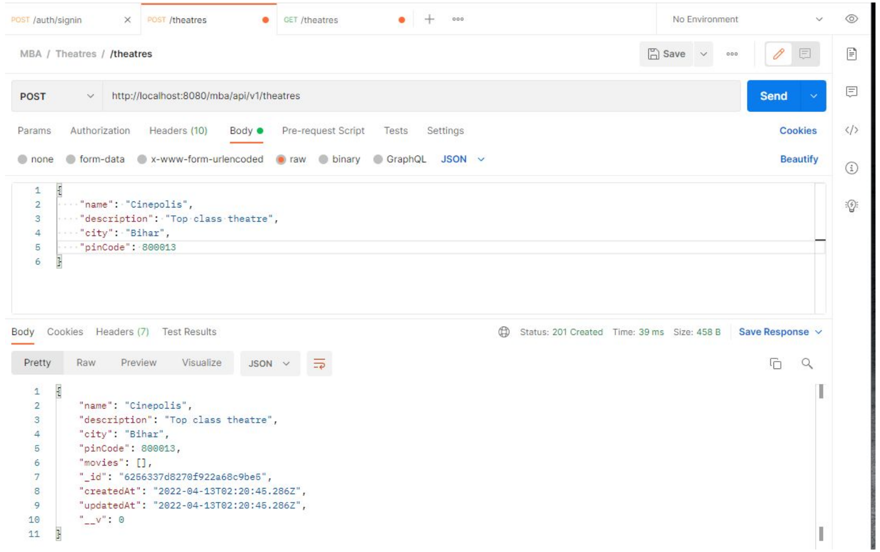


http://localhost:8080/mba/api/v1/theatres

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImF...	
	Key	Value	Description

POST /mba/api/v1/theatres

- The request body is as shown in this image.
- We have received a successful response here.
- Note: we can get the token by creating an account and signing in using signup and signin routes



POST http://localhost:8080/mba/api/v1/theatres

```
{
    "name": "Cinepolis",
    "description": "Top class theatre",
    "city": "Bihar",
    "pinCode": 800013
}
```

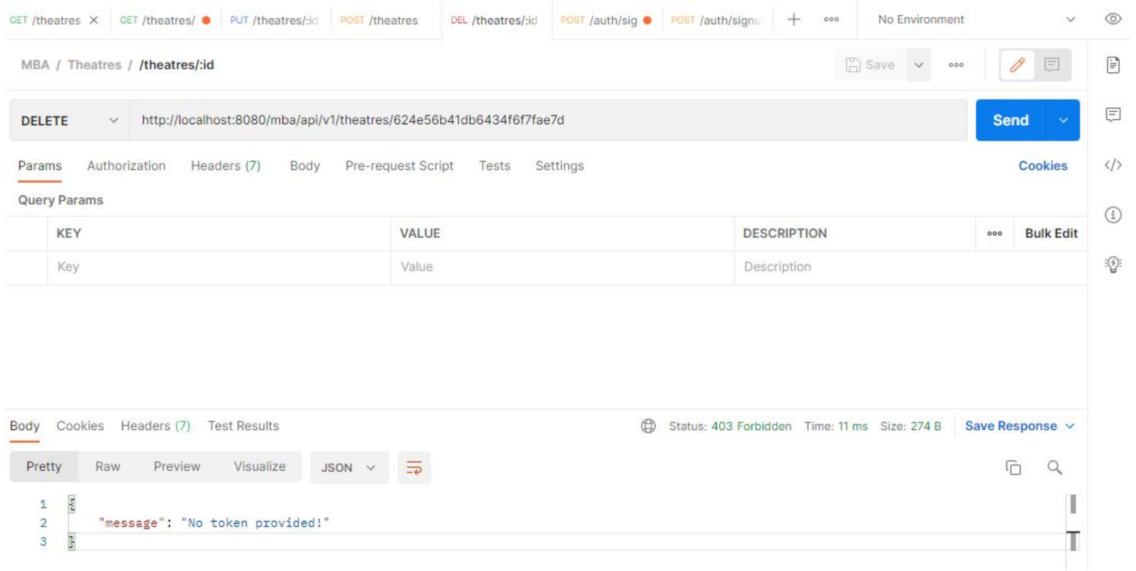
Status: 201 Created Time: 39 ms Size: 458 B

```
{
    "name": "Cinepolis",
    "description": "Top class theatre",
    "city": "Bihar",
    "pinCode": 800013,
    "movies": [],
    "_id": "6256337d8270f922a68c9be5",
    "createdAt": "2022-04-13T02:20:45.286Z",
    "updatedAt": "2022-04-13T02:20:45.286Z",
    "__v": 0
}
```

DELETE /mba/api/v1/theatres/:id

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



DELETE http://localhost:8080/mba/api/v1/theatres/624e56b41db6434f6f7fae7d

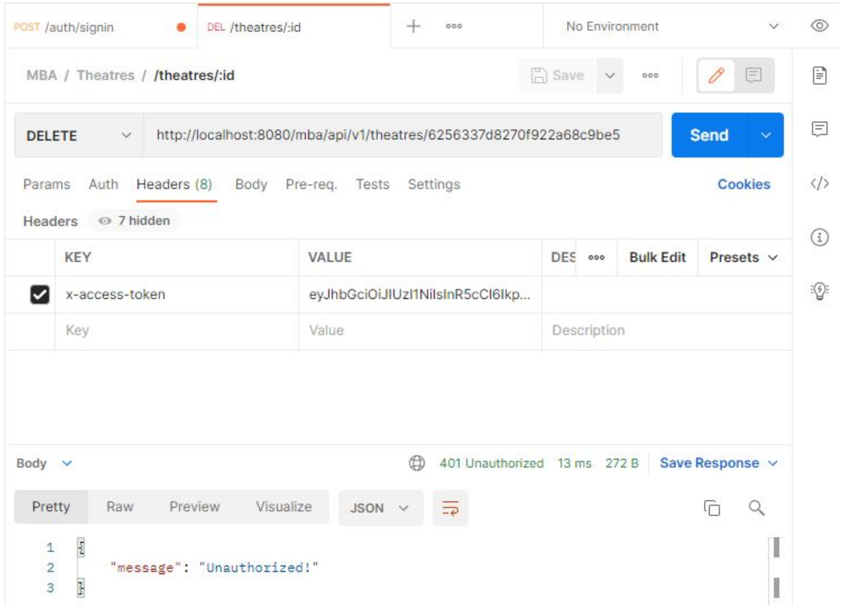
Status: 403 Forbidden Time: 11 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

DELETE /mba/api/v1/theatres/:id

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is sent.



DELETE http://localhost:8080/mba/api/v1/theatres/6256337d8270f922a68c9be5

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6Ikp...	
	Key	Value	Description

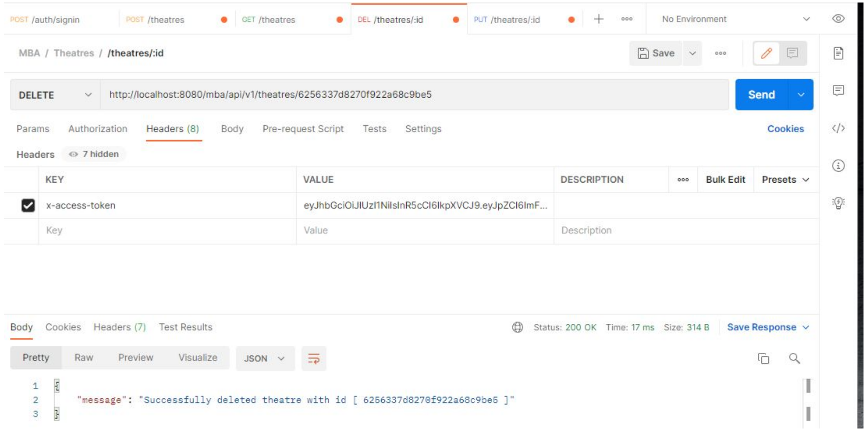
401 Unauthorized 13 ms 272 B

```
{
    "message": "Unauthorized!"
}
```

DELETE /mba/api/v1/theatres/:id

With a token

- Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.
- We have received a successful response here.
- Note: we can get the token by creating an account and signing in using signup and signin routes



DELETE http://localhost:8080/mba/api/v1/theatres/6256337d8270f922a68c9be5

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImF...	
	Key	Value	Description

Status: 200 OK Time: 17 ms Size: 314 B

```
{
    "message": "Successfully deleted theatre with id [ 6256337d8270f922a68c9be5 ]"
}
```

Add authentication and authorization on admin APIs:

- So far we have learned how to protect a route using authentication and validation of token.
- Let us now understand how we can also make some routes accessible only to admin users.
- Admin users are those users who has a role type defined as “ADMIN”
- We will start step by step.

Add authentication and authorization on admin APIs:

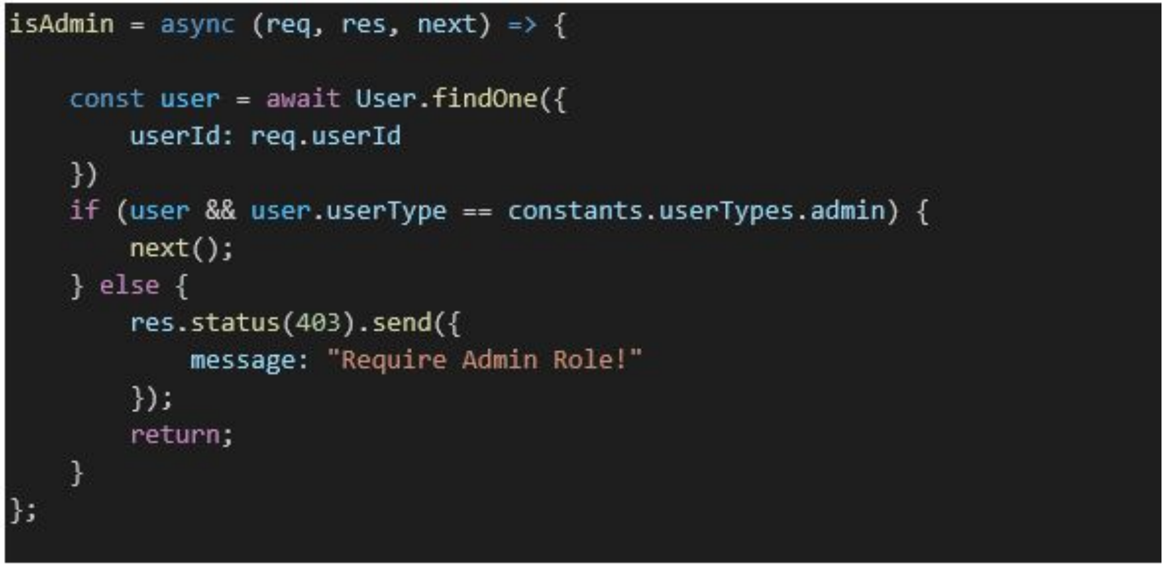
Step 1: Creating Middleware:

Code link:

https://github.com/Vishwa07dev/mba_backend/blob/session6/middlewares/authjwt.js

- isAdmin function here is responsible for validating if a user is having ADMIN access or not.
- First it uses the userId stored in request in verifyToken middleware to find a existing user from the database.
- Next we check if we get a user and the user has a userType of ADMIN, if these are true we proceed for the next function.
- If those check fails it sends a response of "Require Admin Role" with a status code of 403 Unauthorized.

Add authentication and authorization on admin APIs:



```
isAdmin = async (req, res, next) => {

    const user = await User.findOne({
        userId: req.userId
    })
    if (user && user.userType == constants.userTypes.admin) {
        next();
    } else {
        res.status(403).send({
            message: "Require Admin Role!"
        });
        return;
    }
};
```

Add authentication and authorization on admin APIs:

Step 2: We will no go through each resource routes i.e. movies, theatres, users one by one and add the middleware for verification of admin role for specific routes and test those accordingly.

Admins can Create/Update/Delete any movies.

- In Movie Resources we want to have only GET request accessible to CUSTOMER level user.
- While the PUT, DELETE and POST routes should only be accessible to ADMIN level users.
- As the application design suggests to make ADMIN users only to be responsible for creating, updating and deleting a movie from the database.

Admins can Create/Update/Delete any movies.

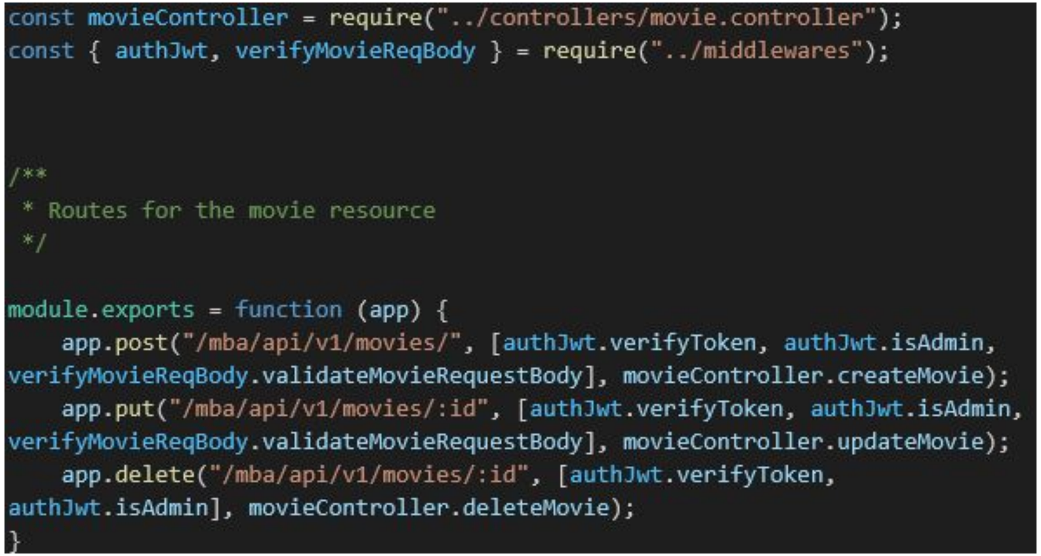
Adding middleware:

Code link:

https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/movie.routes.js

- Here, as we can see the code isAdmin middleware is added in PUT, POST and DELETE routes.
- So that only admins can access these routes.

Admins can Create/Update/Delete any movies.



```
const movieController = require("../controllers/movie.controller");
const { authJwt, verifyMovieReqBody } = require("../middlewares");



/**
 * Routes for the movie resource
 */

module.exports = function (app) {
    app.post("/mba/api/v1/movies/", [authJwt.verifyToken, authJwt.isAdmin,
verifyMovieReqBody.validateMovieRequestBody], movieController.createMovie);
    app.put("/mba/api/v1/movies/:id", [authJwt.verifyToken, authJwt.isAdmin,
verifyMovieReqBody.validateMovieRequestBody], movieController.updateMovie);
    app.delete("/mba/api/v1/movies/:id", [authJwt.verifyToken,
authJwt.isAdmin], movieController.deleteMovie);
}
```

Testing the routes:

Let us now test the APIs by sending a request :

1. With a user access token
2. With an admin access token

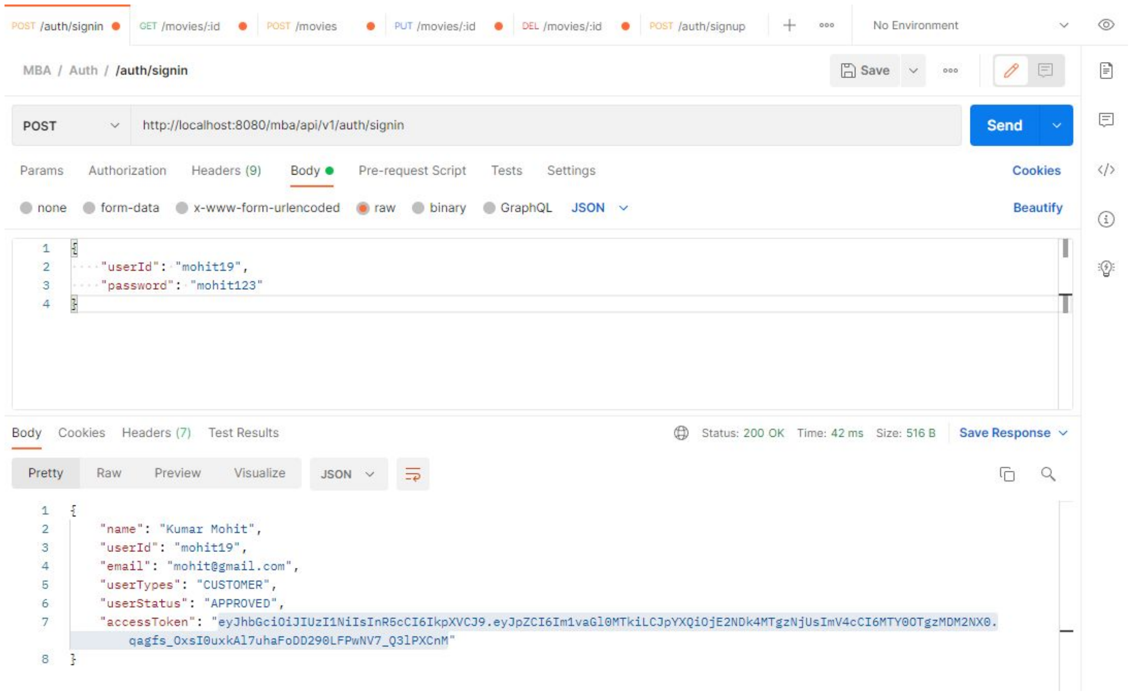
And analyse the result.

Testing the routes:

Before starting the testing let us see in brief step to create a user level token and an admin level token.

1. **For user level token:**
   We need to register a user with signup route and the use signin route to get the token.
2. **For admin level token:**
   We use signin route with admin creds available in server.js file to get the token.

Getting a user token:


POST /auth/signin

MBA / Auth / /auth/signin

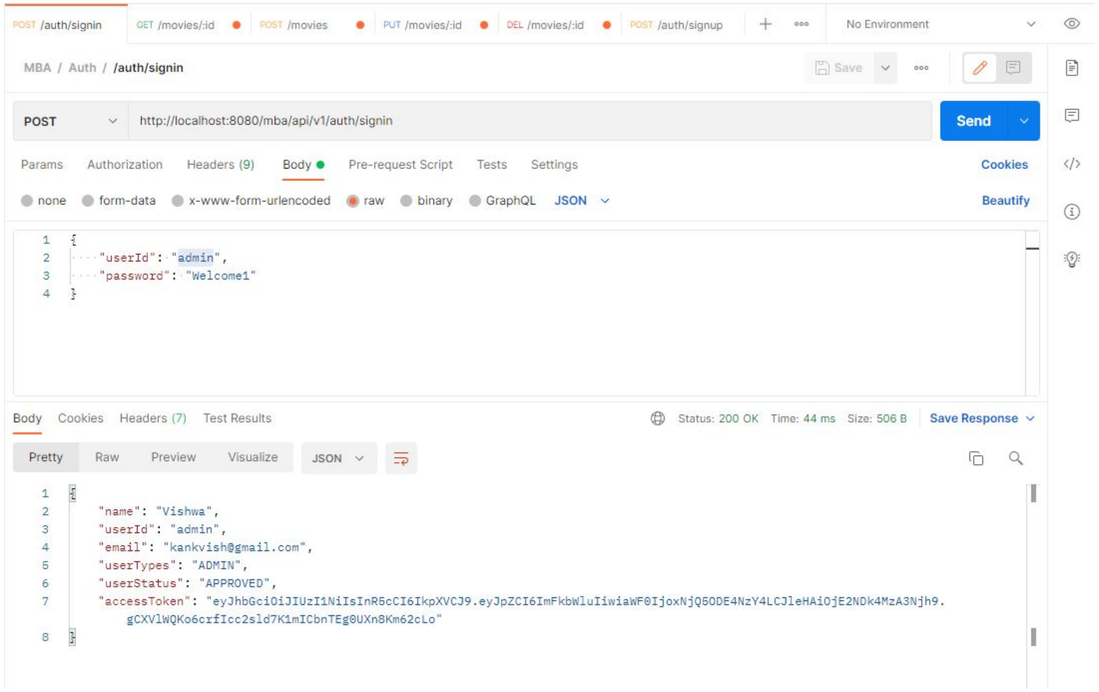
POST http://localhost:8080/mba/api/v1/auth/signin

```
{
    "userId": "mohit19",
    "password": "mohit123"
}
```

Status: 200 OK Time: 42 ms Size: 516 B

```
{
    "name": "Kumar Mohit",
    "userId": "mohit19",
    "email": "mohit@gmail.com",
    "userTypes": "CUSTOMER",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1vaGl0MTkiLCJpYXQiOjE2NDk4MTgzNjUsImV4cCI6MTY0OTgzMDM2NX0.
        qagfs_OxsI0uxkAl7uhaFoDD290LFPwNV7_Q3lPXCnM"
}
```

Getting an Admin token:



POST http://localhost:8080/mba/api/v1/auth/signin

Body: none, form-data, x-www-form-urlencoded, [✓ raw], binary, GraphQL — JSON

```
{
    "userId": "admin",
    "password": "Welcome1"
}
```

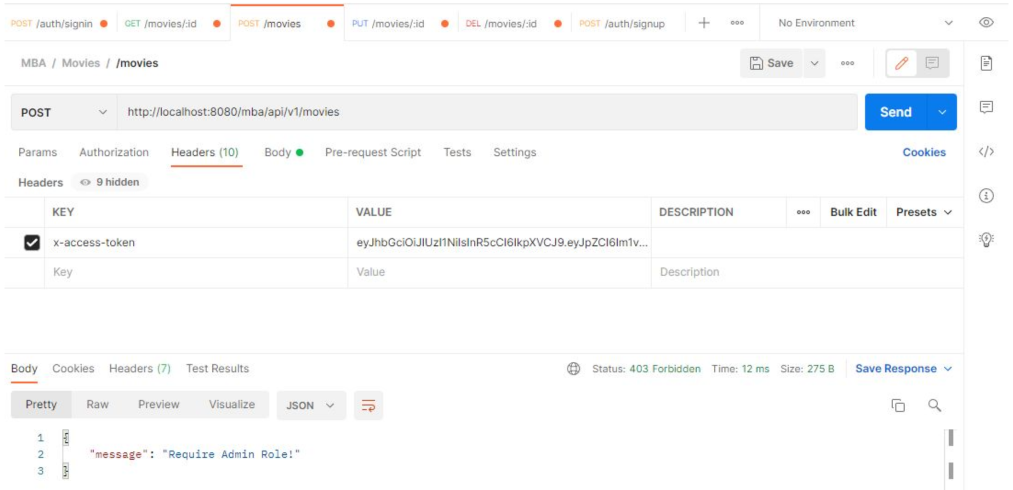
Body — Status: 200 OK Time: 44 ms Size: 506 B — JSON

```
{
    "name": "Vishwa",
    "userId": "admin",
    "email": "kankvish@gmail.com",
    "userTypes": "ADMIN",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImFkbWluIiwiaWF0IjoxNjQ5ODE4NzY4LCJleHAiOjE2NDk4MzA3Njh9.
        gCXVlWQKo6crfIcc2sld7K1mICbnTEg0UXn8Km62cLo"
}
```

POST /mba/api/v1/movies

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



POST http://localhost:8080/mba/api/v1/movies

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

Status: 403 Forbidden Time: 12 ms Size: 275 B

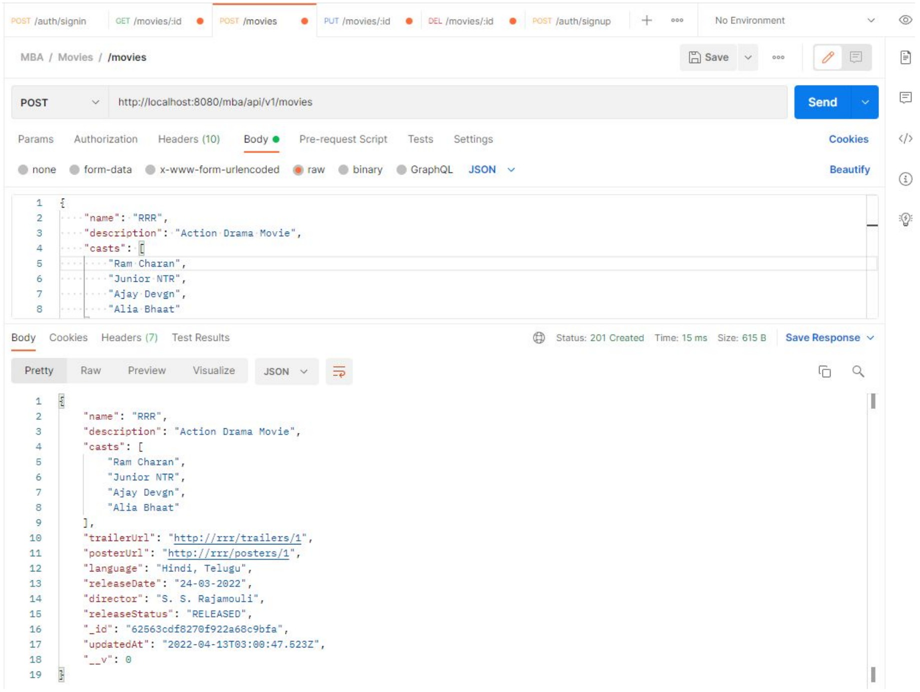
```
{
    "message": "Require Admin Role!"
}
```

POST /mba/api/v1/movies

With an admin token:

- Here, we have passed an admin level token inside request header with the same name "x-access-token".
- And, also the required body for the request.
- We can see a successful response with 200 status code.

POST /mba/api/v1/movies



POST http://localhost:8080/mba/api/v1/movies

```
{
    "name": "RRR",
    "description": "Action Drama Movie",
    "casts": [
        "Ram Charan",
        "Junior NTR",
        "Ajay Devgn",
        "Alia Bhaat"
```

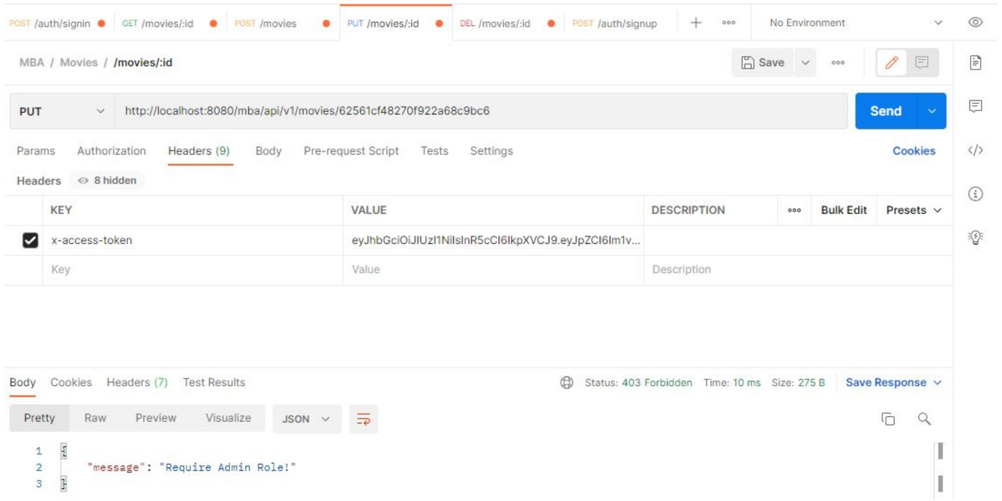
Status: 201 Created Time: 15 ms Size: 615 B

```
{
    "name": "RRR",
    "description": "Action Drama Movie",
    "casts": [
        "Ram Charan",
        "Junior NTR",
        "Ajay Devgn",
        "Alia Bhaat"
    ],
    "trailerUrl": "http://rrr/trailers/1",
    "posterUrl": "http://rrr/posters/1",
    "language": "Hindi, Telugu",
    "releaseDate": "24-03-2022",
    "director": "S. S. Rajamouli",
    "releaseStatus": "RELEASED",
    "_id": "62563cdf8270f922a68c9bfa",
    "updatedAt": "2022-04-13T03:00:47.523Z",
    "__v": 0
}
```

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

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



PUT http://localhost:8080/mba/api/v1/movies/62561cf48270f922a68c9bc6

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

Status: 403 Forbidden Time: 10 ms Size: 275 B

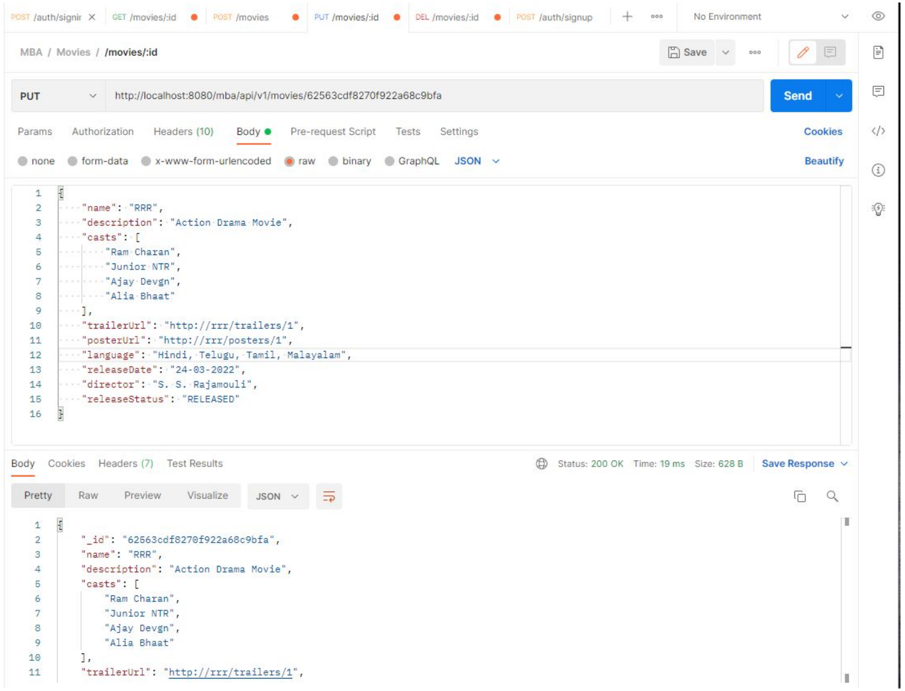
```
{
    "message": "Require Admin Role!"
}
```

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

With an admin token:

- Here, we have passed an admin level token inside request header with the same name "x-access-token".
- And, also the required body for the request.
- We can see a successful response with 200 status code.

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



POST /auth/signin · GET /movies/:id · POST /movies · PUT /movies/:id · DEL /movies/:id · POST /auth/signup · No Environment

MBA / Movies / /movies/:id

PUT http://localhost:8080/mba/api/v1/movies/62563cdf8270f922a68c9bfa

Params · Authorization · Headers (10) · Body · Pre-request Script · Tests · Settings · Cookies

none · form-data · x-www-form-urlencoded · raw · binary · GraphQL · JSON · Beautify

```
{
    "name": "RRR",
    "description": "Action Drama Movie",
    "casts": [
        "Ram Charan",
        "Junior NTR",
        "Ajay Devgn",
        "Alia Bhaat"
    ],
    "trailerUrl": "http://rrr/trailers/1",
    "posterUrl": "http://rrr/posters/1",
    "language": "Hindi, Telugu, Tamil, Malayalam",
    "releaseDate": "24-03-2022",
    "director": "S. S. Rajamouli",
    "releaseStatus": "RELEASED"
}
```

Body · Cookies · Headers (7) · Test Results · Status: 200 OK · Time: 19 ms · Size: 628 B · Save Response

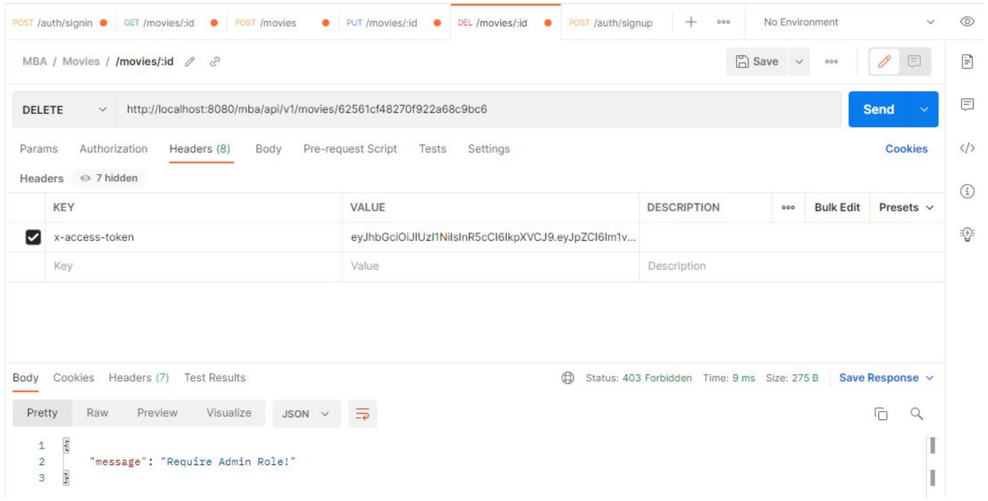
Pretty · Raw · Preview · Visualize · JSON

```
{
    "_id": "62563cdf8270f922a68c9bfa",
    "name": "RRR",
    "description": "Action Drama Movie",
    "casts": [
        "Ram Charan",
        "Junior NTR",
        "Ajay Devgn",
        "Alia Bhaat"
    ],
    "trailerUrl": "http://rrr/trailers/1",
```

DELETE /mba/api/v1/movies/:id

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



DELETE http://localhost:8080/mba/api/v1/movies/62561cf48270f922a68c9bc6

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

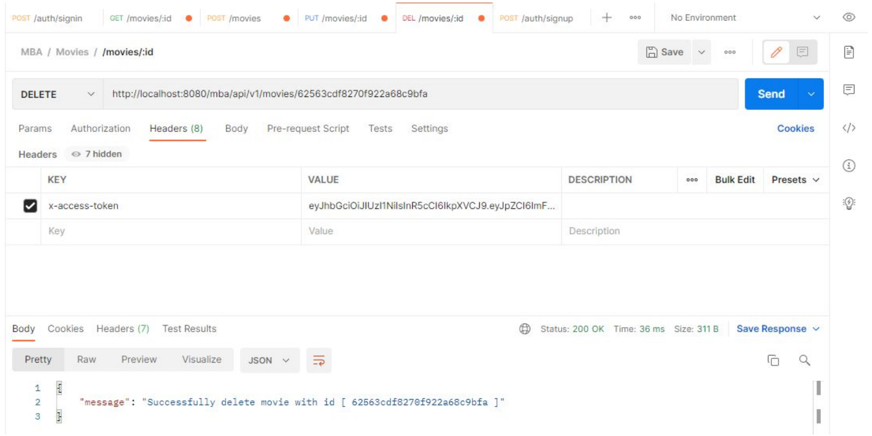
Status: 403 Forbidden Time: 9 ms Size: 275 B

```
{
    "message": "Require Admin Role!"
}
```

DELETE /mba/api/v1/movies/:id

With an admin token:

- Here, we have passed an admin level token inside request header with the same name “x-access-token”.
- And, also the required body for the request.
- We can see a successful response with 200 status code.



POST /auth/signin | GET /movies/:id | POST /movies | PUT /movies/:id | DEL /movies/:id | POST /auth/signup | No Environment

MBA / Movies / /movies/:id

DELETE http://localhost:8080/mba/api/v1/movies/62563cdf8270f922a68c9bfa Send

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

Headers 7 hidden

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImF...	
	Key	Value	Description

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 36 ms Size: 311 B Save Response

Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "Successfully delete movie with id [ 62563cdf8270f922a68c9bfa ]"
}
```

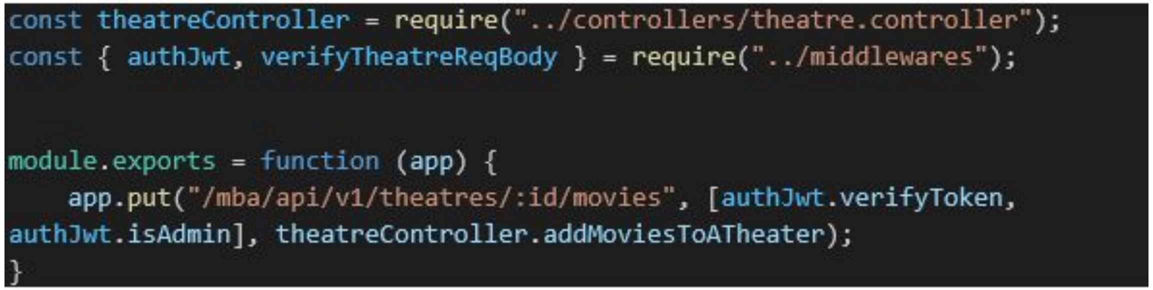
Admins can Create/Update/Delete any movies in any theatre

- /mba/api/v1/theatres/:id/movies is used to insert or remove any movie from a theatre.
- This route also qualifies for a API which should only be accessible for ADMIN user to handle.
- So, we will also add ADMIN validation middleware for this route.

Admins can Create/Update/Delete any movies in any theatre

Adding middleware:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/theatre.routes.js

- Here, as we can see the code isAdmin middleware is added in PUT route.
- So that only admins can access this route.



```
const theatreController = require("../controllers/theatre.controller");
const { authJwt, verifyTheatreReqBody } = require("../middlewares");


module.exports = function (app) {
    app.put("/mba/api/v1/theatres/:id/movies", [authJwt.verifyToken,
authJwt.isAdmin], theatreController.addMoviesToATheater);
}
```

Testing the route:

Let us now test the APIs by sending a request :

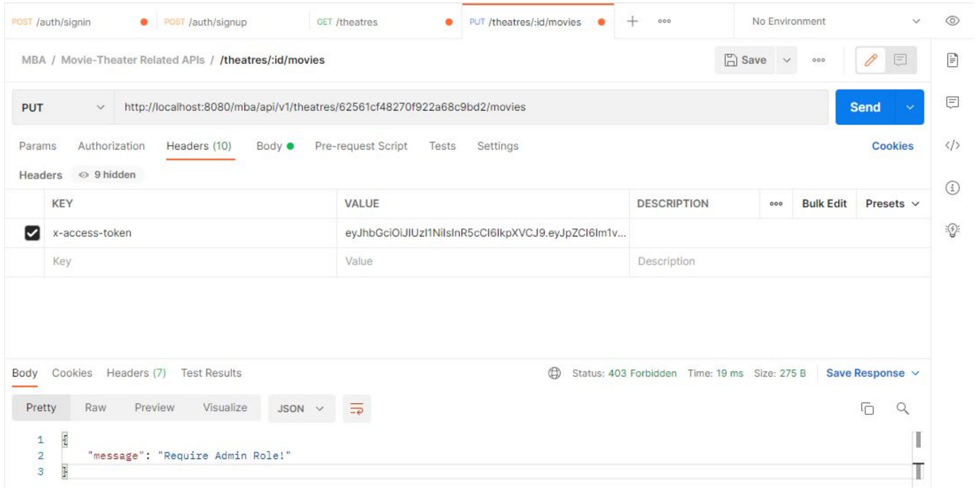
1. With a user access token
2. With an admin access token

And analyse the result.

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

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



PUT http://localhost:8080/mba/api/v1/theatres/62561cf48270f922a68c9bd2/movies

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

Status: 403 Forbidden Time: 19 ms Size: 275 B

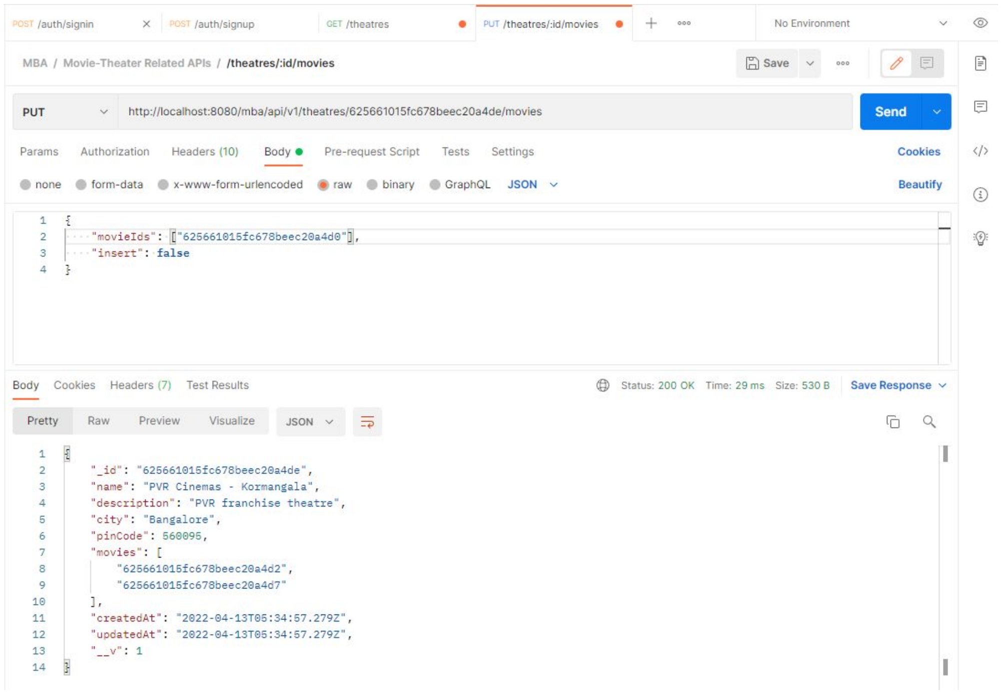
```
{
    "message": "Require Admin Role!"
}
```

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

With an admin token:

- Here, we have passed an admin level token inside request header with the same name "x-access-token".
- And, also the required body for the request.
- We can see a successful response with 200 status code.

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


POST /auth/signin
POST /auth/signup
GET /theatres
PUT /theatres/:id/movies
No Environment
MBA / Movie-Theater Related APIs / /theatres/:id/movies
Save
Send
Params Authorization Headers (10) Body Pre-request Script Tests Settings
Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON
Beautify
Body Cookies Headers (7) Test Results
Save Response
Pretty Raw Preview Visualize JSON

PUT http://localhost:8080/mba/api/v1/theatres/625661015fc678beec20a4de/movies

```
{
    "movieIds": ["625661015fc678beec20a4d0"],
    "insert": false
}
```

Status: 200 OK Time: 29 ms Size: 530 B

```
{
    "_id": "625661015fc678beec20a4de",
    "name": "PVR Cinemas - Kormangala",
    "description": "PVR franchise theatre",
    "city": "Bangalore",
    "pinCode": 560095,
    "movies": [
        "625661015fc678beec20a4d2",
        "625661015fc678beec20a4d7"
    ],
    "createdAt": "2022-04-13T05:34:57.279Z",
    "updatedAt": "2022-04-13T05:34:57.279Z",
    "__v": 1
}
```

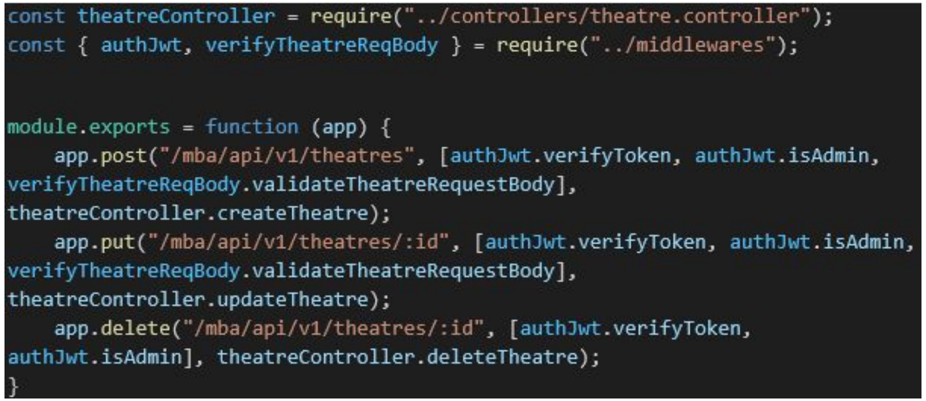
Admins can Create/Update/Delete any theatres:

Adding middleware:

Code link:

https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/theatre.routes.js

- Here, as we can see the code isAdmin middleware is added in PUT, POST and DELETE routes.
- So that only admins can access these routes.



```
const theatreController = require("../controllers/theatre.controller");
const { authJwt, verifyTheatreReqBody } = require("../middlewares");


module.exports = function (app) {
    app.post("/mba/api/v1/theatres", [authJwt.verifyToken, authJwt.isAdmin,
verifyTheatreReqBody.validateTheatreRequestBody],
theatreController.createTheatre);
    app.put("/mba/api/v1/theatres/:id", [authJwt.verifyToken, authJwt.isAdmin,
verifyTheatreReqBody.validateTheatreRequestBody],
theatreController.updateTheatre);
    app.delete("/mba/api/v1/theatres/:id", [authJwt.verifyToken,
authJwt.isAdmin], theatreController.deleteTheatre);
}
```

Testing the route:

Let us now test the APIs by sending a request :

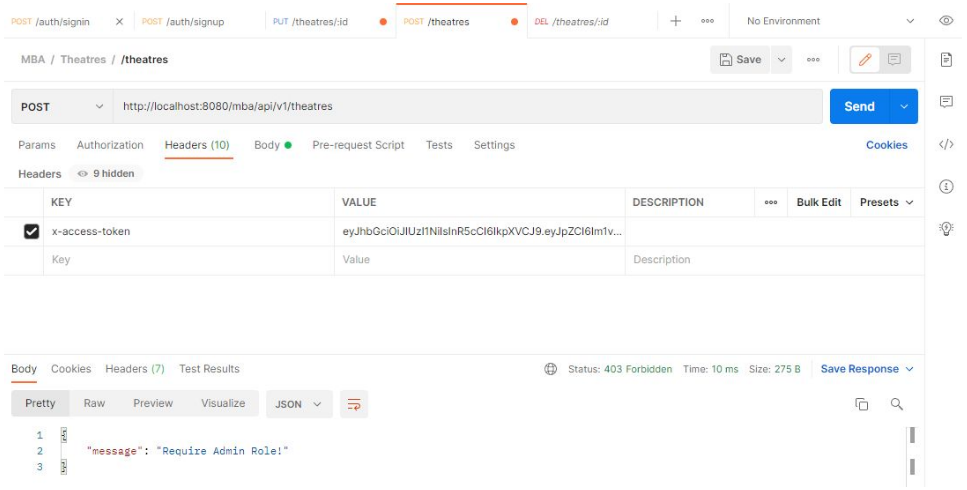
1. With a user access token
2. With an admin access token

And analyse the result.

POST /mba/api/v1/theatres

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



POST http://localhost:8080/mba/api/v1/theatres

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

Status: 403 Forbidden Time: 10 ms Size: 275 B

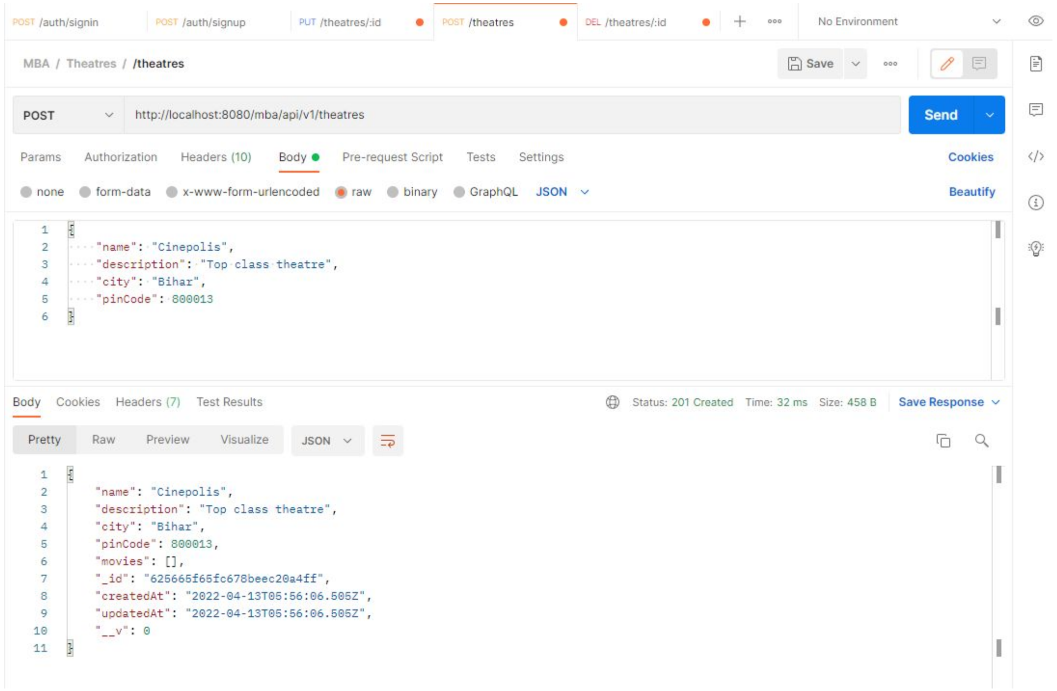
```
{
    "message": "Require Admin Role!"
}
```

POST /mba/api/v1/theatres

With an admin token:

- Here, we have passed an admin level token inside request header with the same name “x-access-token”.
- And, also the required body for the request.
- We can see a successful response with 200 status code.

POST /mba/api/v1/theatres



POST http://localhost:8080/mba/api/v1/theatres

```
{
    "name": "Cinepolis",
    "description": "Top class theatre",
    "city": "Bihar",
    "pinCode": 800013
}
```

Status: 201 Created Time: 32 ms Size: 458 B

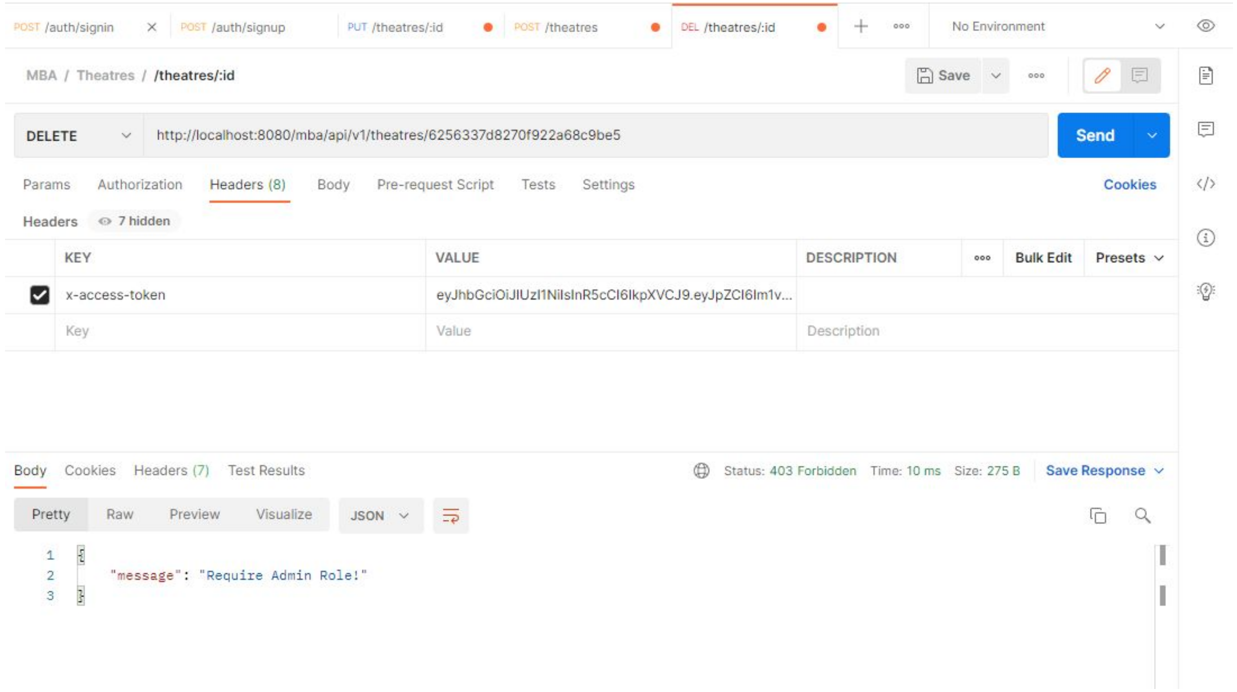
```
{
    "name": "Cinepolis",
    "description": "Top class theatre",
    "city": "Bihar",
    "pinCode": 800013,
    "movies": [],
    "_id": "625665f65fc678beec20a4ff",
    "createdAt": "2022-04-13T05:56:06.505Z",
    "updatedAt": "2022-04-13T05:56:06.505Z",
    "__v": 0
}
```

DELETE /mba/api/v1/theatres/:id

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.

DELETE /mba/api/v1/theatres/:id



DELETE http://localhost:8080/mba/api/v1/theatres/6256337d8270f922a68c9be5

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

Status: 403 Forbidden Time: 10 ms Size: 275 B

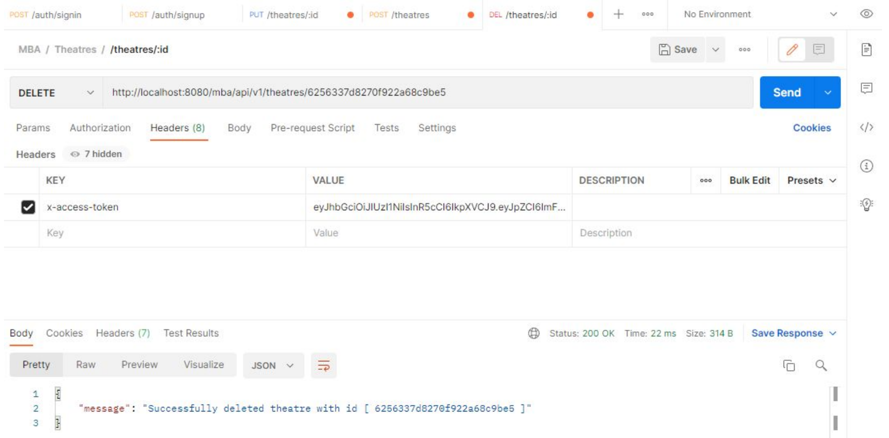
```
{
    "message": "Require Admin Role!"
}
```

BED-Class #180DaysofPurpose Relevel by Unacademy

DELETE /mba/api/v1/theatres/:id

With an admin token:

- Here, we have passed an admin level token inside request header with the same name "x-access-token".
- And, also the required body for the request.
- We can see a successful response with 200 status code.



POST /auth/signin | POST /auth/signup | PUT /theatres/:id | POST /theatres | DEL /theatres/:id | No Environment

MBA / Theatres / /theatres/:id

DELETE http://localhost:8080/mba/api/v1/theatres/6256337d8270f922a68c9be5

Params Authorization Headers (8) Body Pre-request Script Tests Settings Cookies

Headers 7 hidden

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6ImF...	
	Key	Value	Description

Body Cookies Headers (7) Test Results — Status: 200 OK Time: 22 ms Size: 314 B

Pretty Raw Preview Visualize JSON

```json
{
    "message": "Successfully deleted theatre with id [ 6256337d8270f922a68c9be5 ]"
}
```

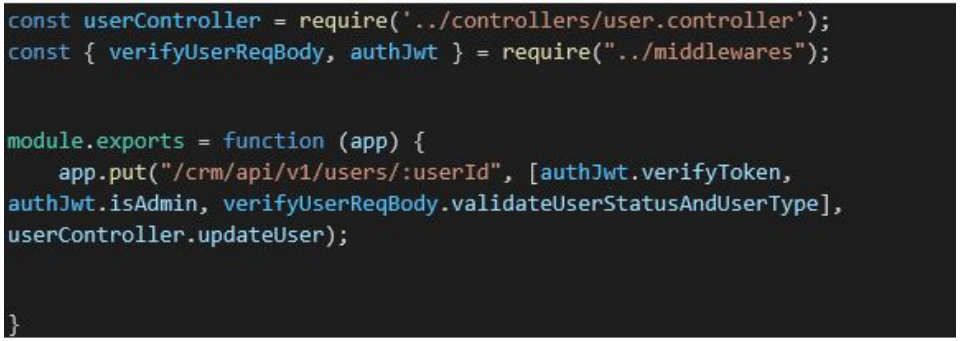
Admin can update the details of any type of users:

- PUT /crm/api/v1/users/:userId : this endpoint is made for a purpose to accept CLIENT level users and other ADMIN for the application.
- So, to protect this route we have to add the middleware to verify ADMIN level users only.

Admin can update the details of any type of users:

Adding middleware:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/user.routes.js

- Here, as we can see the code isAdmin middleware is added in PUT route.
- So that only admins can access this route.



```
const userController = require('../controllers/user.controller');
const { verifyUserReqBody, authJwt } = require("../middlewares");


module.exports = function (app) {
    app.put("/crm/api/v1/users/:userId", [authJwt.verifyToken,
authJwt.isAdmin, verifyUserReqBody.validateUserStatusAndUserType],
userController.updateUser);


}
```

Testing the routes

Let us now test the APIs by sending a request :

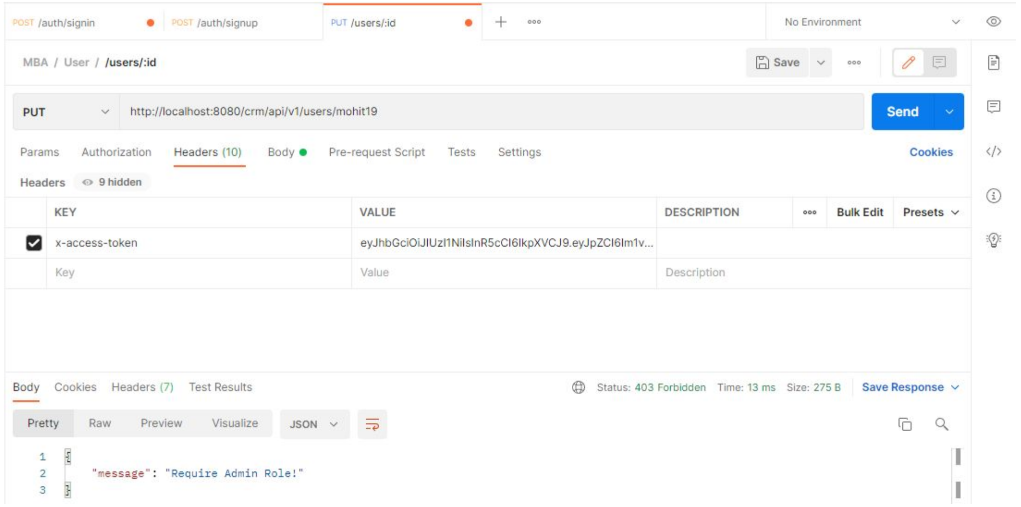
1. With a user access token
2. With an admin access token

And analyse the result.

PUT /crm/api/v1/users/:userId:

Without an admin token:

- Here, we have passed a user level token in the request header as “x-access-token”.
- We can see the response here is “Require Admin role” with forbidden status code of 403.
- This is as expected response.



POST /auth/signin · POST /auth/signup · PUT /users/:id · No Environment

MBA / User / /users/:id

PUT http://localhost:8080/crm/api/v1/users/mohit19 · Send

Params · Authorization · Headers (10) · Body · Pre-request Script · Tests · Settings · Cookies

Headers · 9 hidden

	KEY	VALUE	DESCRIPTION
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
	Key	Value	Description

Body · Cookies · Headers (7) · Test Results · Status: 403 Forbidden · Time: 13 ms · Size: 275 B · Save Response

Pretty · Raw · Preview · Visualize · JSON

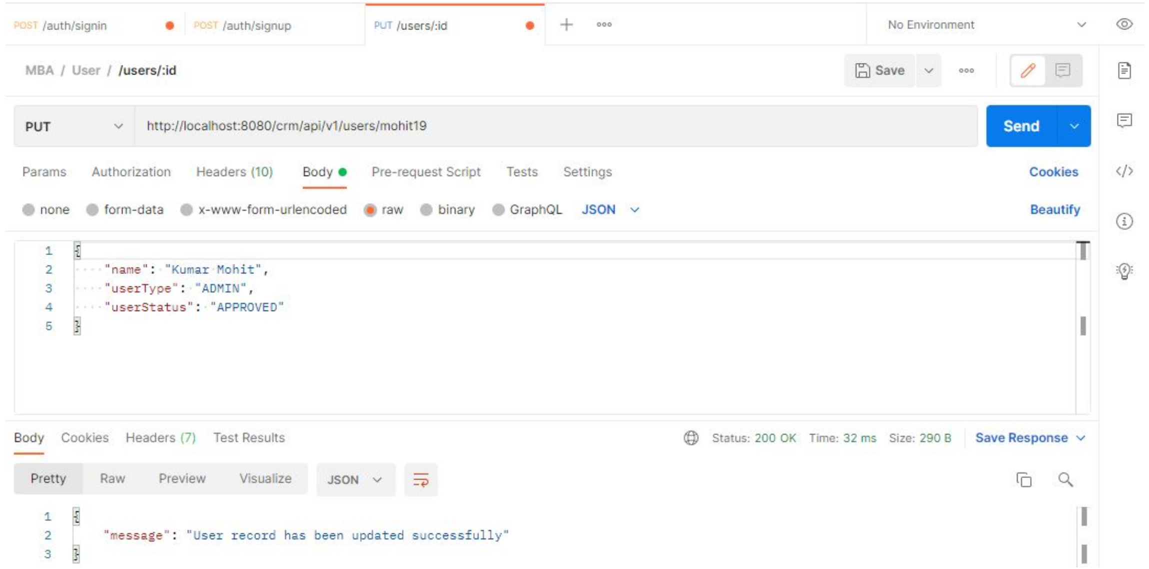
```
{
    "message": "Require Admin Role!"
}
```

PUT /crm/api/v1/users/:userId:

With an admin token:

- Here, we have passed an admin level token inside request header with the same name "x-access-token".
- And, also the required body for the request where we have made a CUSTOMER level user as ADMIN.
- We can see a successful response with 200 status code.

PUT /crm/api/v1/users/:userId:



POST /auth/signin POST /auth/signup PUT /users/:id

No Environment

MBA / User / /users/:id

Save

PUT http://localhost:8080/crm/api/v1/users/mohit19

Send

Params Authorization Headers (10) Body Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
{
    "name": "Kumar Mohit",
    "userType": "ADMIN",
    "userStatus": "APPROVED"
}
```

Body Cookies Headers (7) Test Results Status: 200 OK Time: 32 ms Size: 290 B Save Response

Pretty Raw Preview Visualize JSON

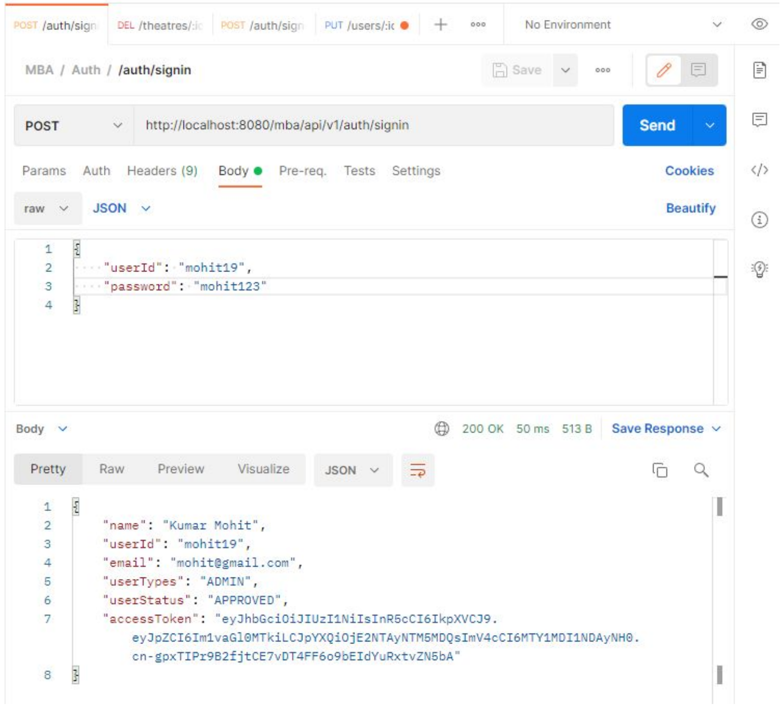
```
{
    "message": "User record has been updated successfully"
}
```

PUT /crm/api/v1/users/:userId:

Signin to verify user status:

- To verify the user granted with ADMIN access let us try to login with the user credentials and confirm.
- As you can see from the response here, now the user as a userType of ADMIN and can also access ADMIN level APIs and privileges.

PUT /crm/api/v1/users/:userId:



POST http://localhost:8080/mba/api/v1/auth/signin

```
{
    "userId": "mohit19",
    "password": "mohit123"
}
```

200 OK 50 ms 513 B

```
{
    "name": "Kumar Mohit",
    "userId": "mohit19",
    "email": "mohit@gmail.com",
    "userTypes": "ADMIN",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6Im1vaGl0MTkiLCJpYXQiOjE2NTAyNTM5MDQsImV4cCI6MTY1MDI1NDAyNH0.
        cn-gpxTIPr9B2fjtCE7vDT4FF6o9bEIdYuRxtvZN5bA"
}
```

Only authenticated user should be allowed to use any other API

- Apart from all the routes we discussed before are only available to ADMIN users.
- Now, let us add user level validation for the rest of the APIs.
- This will make them available to all types of users.
- Only token validation middleware need to be added for these routes.
- Let's look at them one by one.

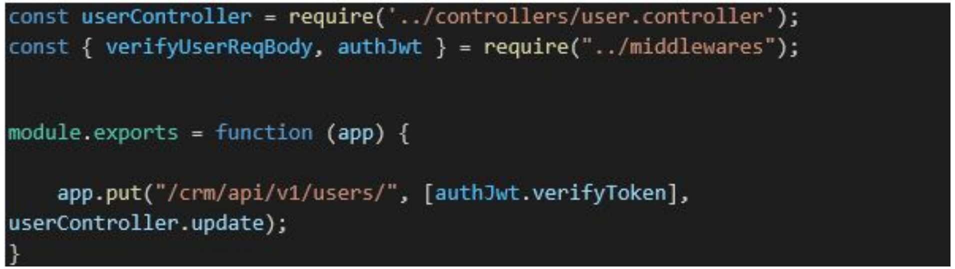
User API:

- PUT /crm/api/v1/users/
- This API is created for users to update their password and should only be available to an authenticated user.
- So, we add verifyToken middleware for this route.

User API:

Adding middleware:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/user.routes.js

- As you can see in from the code route we have added a function in the middle authJwt.verifyToken which as explained earlier validates that the user accessing these routes have a valid token.



```
const userController = require('../controllers/user.controller');
const { verifyUserReqBody, authJwt } = require("../middlewares");


module.exports = function (app) {

    app.put("/crm/api/v1/users/", [authJwt.verifyToken],
userController.update);
}
```

Testing the route:

Let us now test the APIs by sending a request :

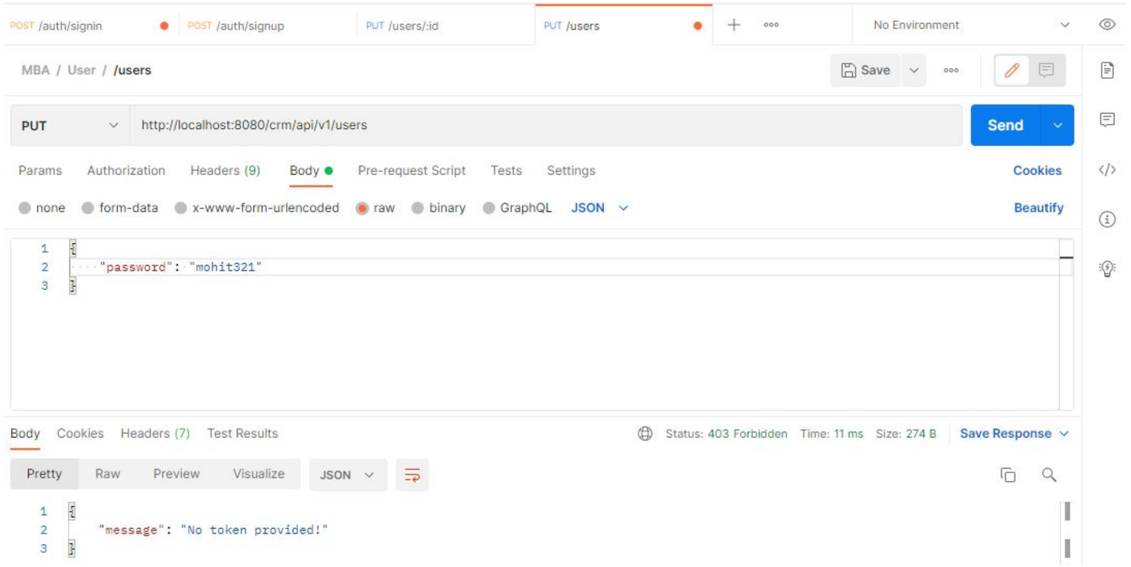
1. Without a token
2. With a token

And analyse the result.

PUT /crm/api/v1/users

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



PUT http://localhost:8080/crm/api/v1/users

```
{
    "password": "mohit321"
}
```

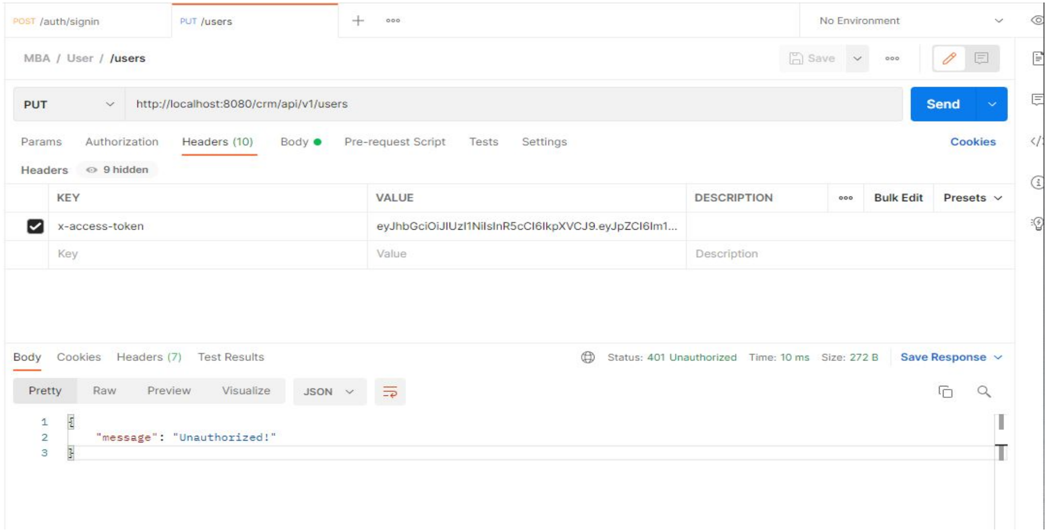
Status: 403 Forbidden Time: 11 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

PUT /crm/api/v1/users

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is recieved.



PUT http://localhost:8080/crm/api/v1/users

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1...	
Key	Value	Description

Status: 401 Unauthorized Time: 10 ms Size: 272 B

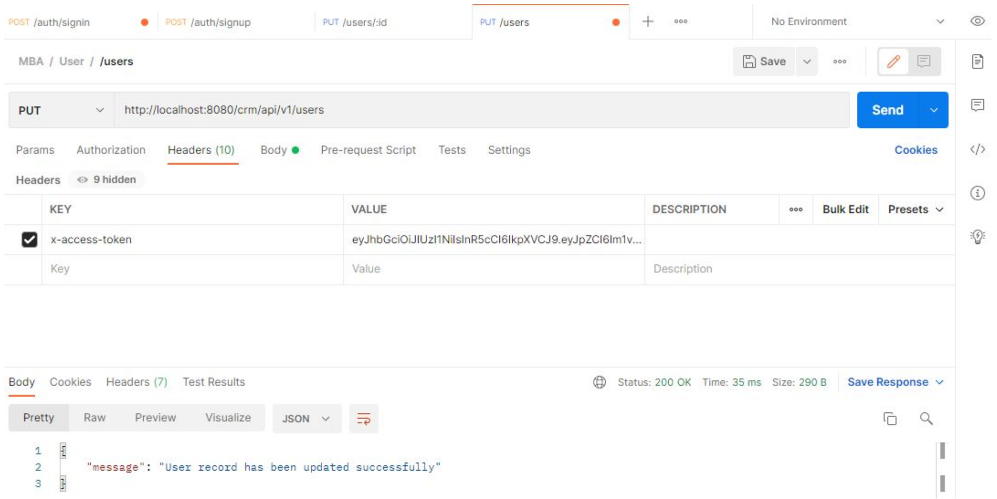
```
{
    "message": "Unauthorized!"
}
```

PUT /crm/api/v1/users

With a token:

Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.

We have received a successful response here.



PUT http://localhost:8080/crm/api/v1/users

KEY	VALUE	DESCRIPTION
x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

Status: 200 OK Time: 35 ms Size: 290 B

```
{
    "message": "User record has been updated successfully"
}
```

BED-Class #180DaysofPurpose Relevel by Unacademy

Theatre API:

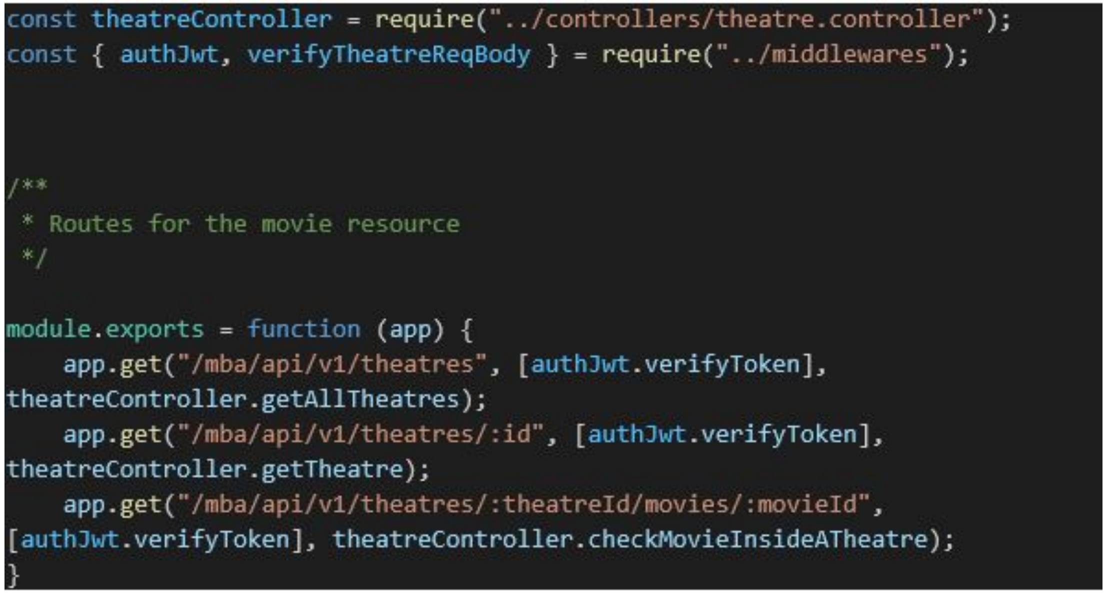
- GET /mba/api/v1/theatres
- GET /mba/api/v1/theatres/:id
- GET /mba/api/v1/theatres/:theatreId/movies/:movieId
- This APIs should be accessible to all level users with a valid token.
- So, we add verifyToken middleware for this route.

Theatre API:

Adding middleware:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/theatre.routes.js

- As you can see in from the code in each route we have added a function in the middle authJwt.verifyToken which as explained earlier validates that the user accessing these routes have a valid token.

Theatre API:



```
const theatreController = require("../controllers/theatre.controller");
const { authJwt, verifyTheatreReqBody } = require("../middlewares");



/**
 * Routes for the movie resource
 */

module.exports = function (app) {
    app.get("/mba/api/v1/theatres", [authJwt.verifyToken],
theatreController.getAllTheatres);
    app.get("/mba/api/v1/theatres/:id", [authJwt.verifyToken],
theatreController.getTheatre);
    app.get("/mba/api/v1/theatres/:theatreId/movies/:movieId",
[authJwt.verifyToken], theatreController.checkMovieInsideATheatre);
}
```

Testing the routes:

- We have already tested these routes earlier in the class while creating a middleware to validate access token.

Movie API:

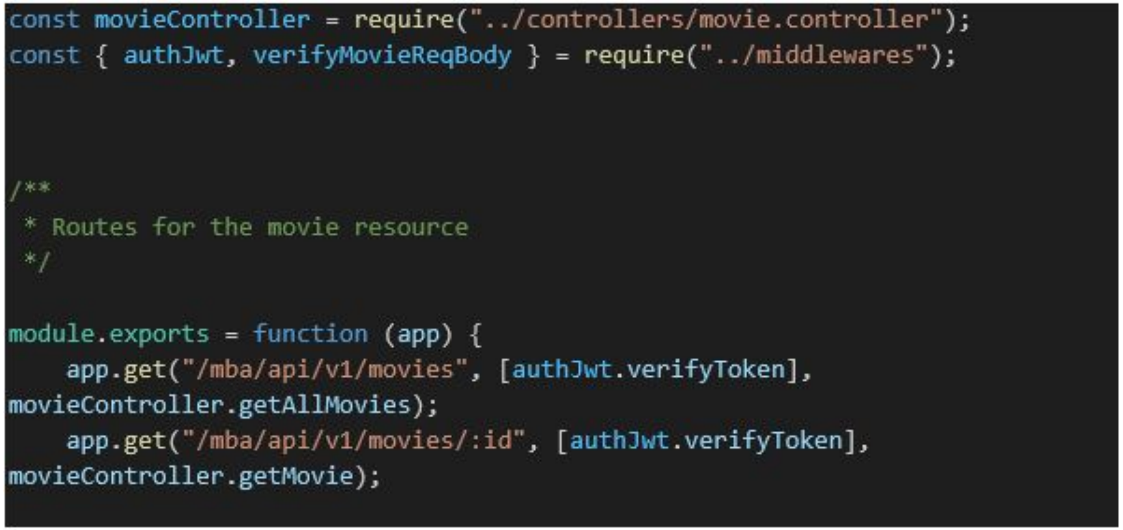
- GET /mba/api/v1/movies
- GET /mba/api/v1/movies/:id
- This APIs should be accessible to all level users with a valid token.
- So, we add verifyToken middleware for this route.

Movie API:

Adding middleware:
Code link:
https://github.com/Vishwa07dev/mba_backend/blob/session6/routes/movie.routes.js

- As you can see in from the code in all the route we have added a function in the middle authJwt.verifyToken which as explained earlier validates that the user accessing these routes have a valid token.

Movie API:



```
const movieController = require("../controllers/movie.controller");
const { authJwt, verifyMovieReqBody } = require("../middlewares");



/**
 * Routes for the movie resource
 */

module.exports = function (app) {
    app.get("/mba/api/v1/movies", [authJwt.verifyToken],
movieController.getAllMovies);
    app.get("/mba/api/v1/movies/:id", [authJwt.verifyToken],
movieController.getMovie);
```

Testing the route:

Let us now test the APIs by sending a request :

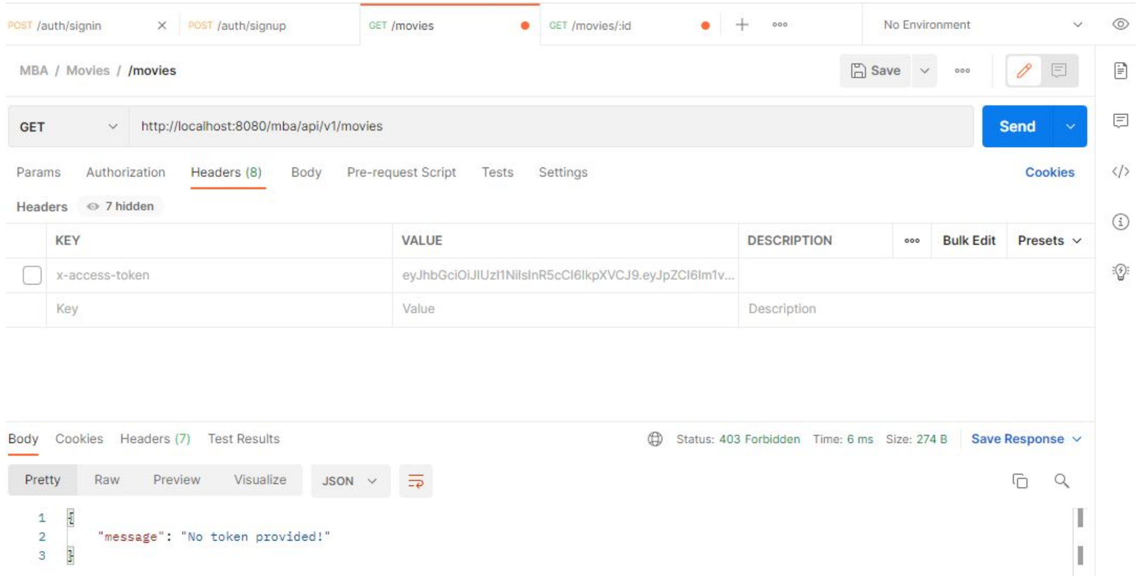
1. Without a token
2. With a token

And analyse the result.

GET /mba/api/v1/movies

Without a token:

Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



POST /auth/signin | POST /auth/signup | GET /movies | GET /movies/:id | No Environment

MBA / Movies / /movies

GET http://localhost:8080/mba/api/v1/movies

Params | Authorization | Headers (8) | Body | Pre-request Script | Tests | Settings | Cookies

Headers 7 hidden

KEY	VALUE	DESCRIPTION
x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

Body | Cookies | Headers (7) | Test Results — Status: 403 Forbidden Time: 6 ms Size: 274 B

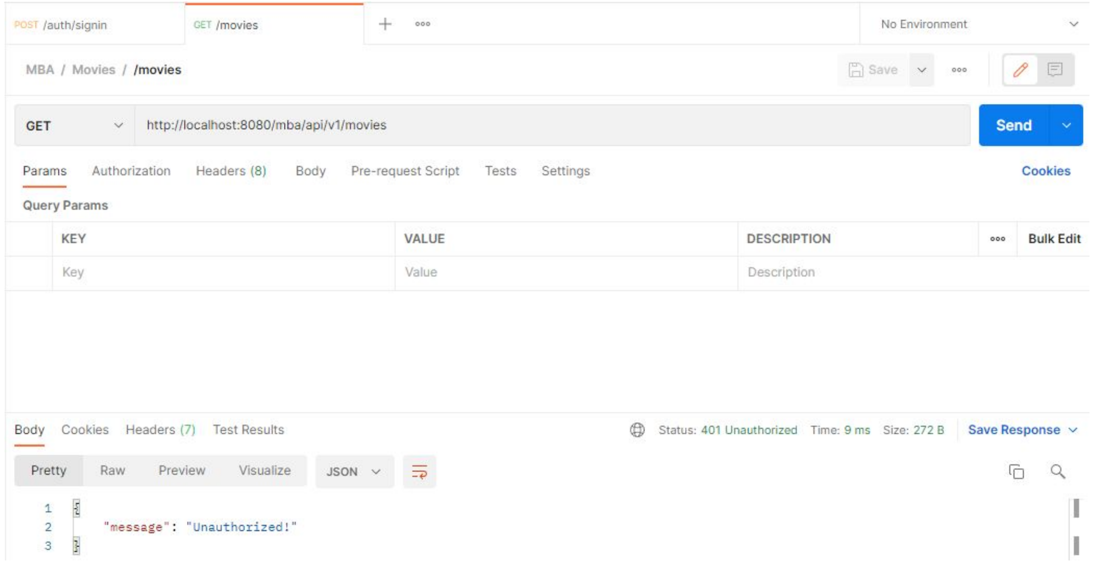
Pretty | Raw | Preview | Visualize | JSON

```
{
    "message": "No token provided!"
}
```

GET /mba/api/v1/movies

With an invalid/ expired token:

When an expired or invalid token is sent a 401 – Unauthorized response is sent.



GET http://localhost:8080/mba/api/v1/movies

KEY	VALUE	DESCRIPTION
Key	Value	Description

Status: 401 Unauthorized Time: 9 ms Size: 272 B

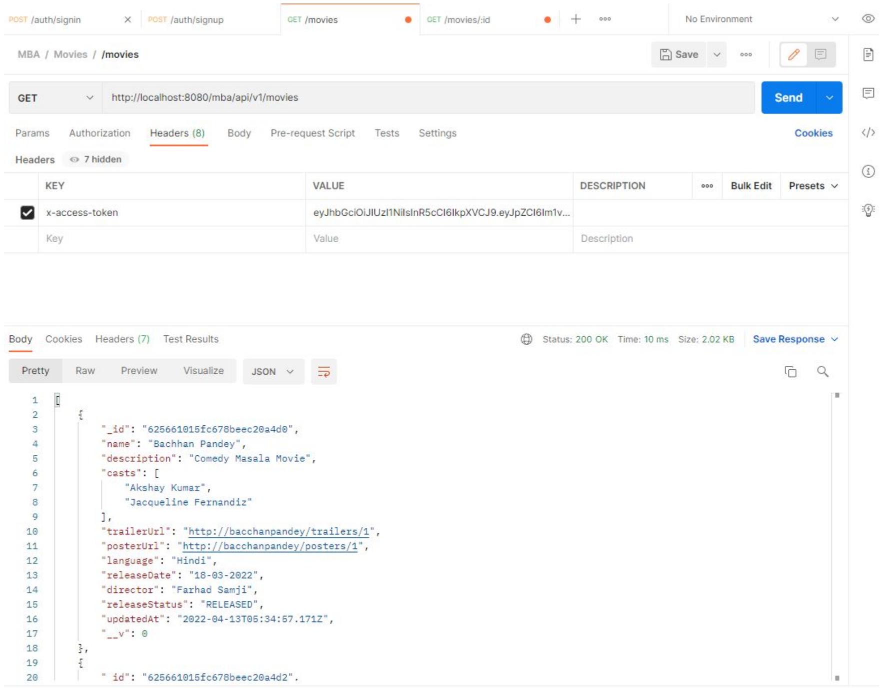
```
{
    "message": "Unauthorized!"
}
```

GET /mba/api/v1/movies

With a token:

- Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.
- We have received a successful response here.

GET /mba/api/v1/movies



GET http://localhost:8080/mba/api/v1/movies

KEY	VALUE	DESCRIPTION
✓ x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

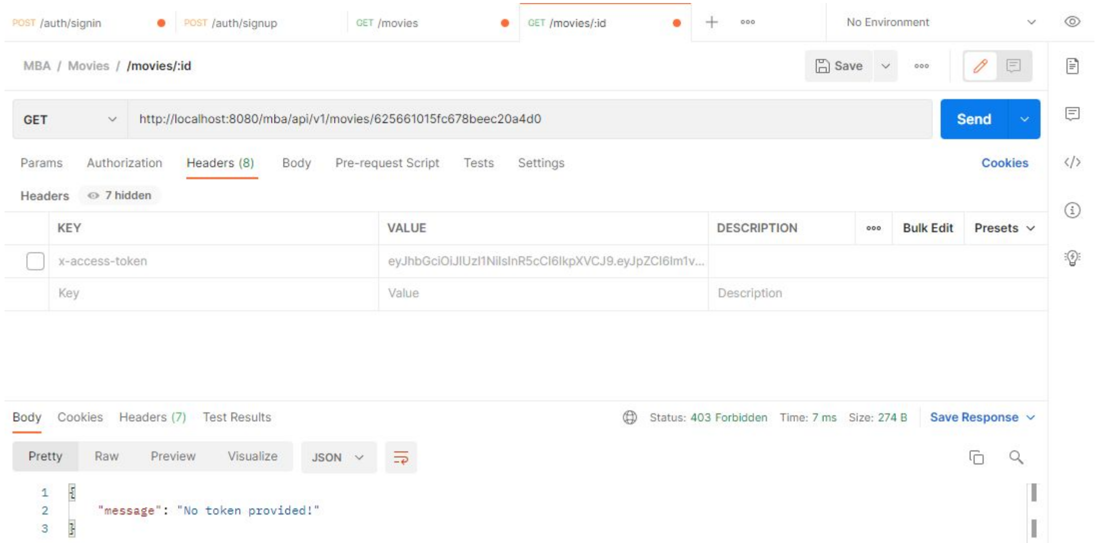
Status: 200 OK Time: 10 ms Size: 2.02 KB

```
[
    {
        "_id": "625661015fc678beec20a4d0",
        "name": "Bachhan Pandey",
        "description": "Comedy Masala Movie",
        "casts": [
            "Akshay Kumar",
            "Jacqueline Fernandiz"
        ],
        "trailerUrl": "http://bacchanpandey/trailers/1",
        "posterUrl": "http://bacchanpandey/posters/1",
        "language": "Hindi",
        "releaseDate": "18-03-2022",
        "director": "Farhad Samji",
        "releaseStatus": "RELEASED",
        "updatedAt": "2022-04-13T05:34:57.171Z",
        "__v": 0
    },
    {
        "_id": "625661015fc678beec20a4d2".
```

GET /mba/api/v1/movies/:id

Without a token:

- Here the output is No token provided with a status code of 403 Forbidden, which is as we expected.



GET http://localhost:8080/mba/api/v1/movies/625661015fc678beec20a4d0

KEY	VALUE	DESCRIPTION
x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	
Key	Value	Description

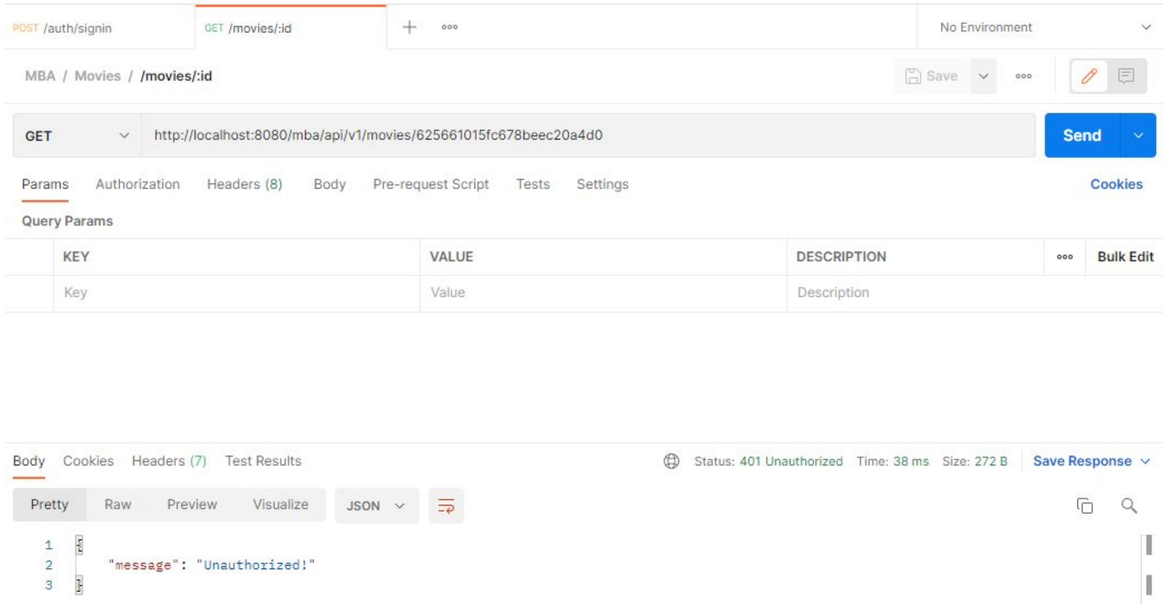
Status: 403 Forbidden Time: 7 ms Size: 274 B

```
{
    "message": "No token provided!"
}
```

GET /mba/api/v1/movies/:id

With an invalid/ expired token:

- When an expired or invalid token is sent a 401 – Unauthorized response is sent.



http://localhost:8080/mba/api/v1/movies/625661015fc678beec20a4d0

Status: 401 Unauthorized Time: 38 ms Size: 272 B

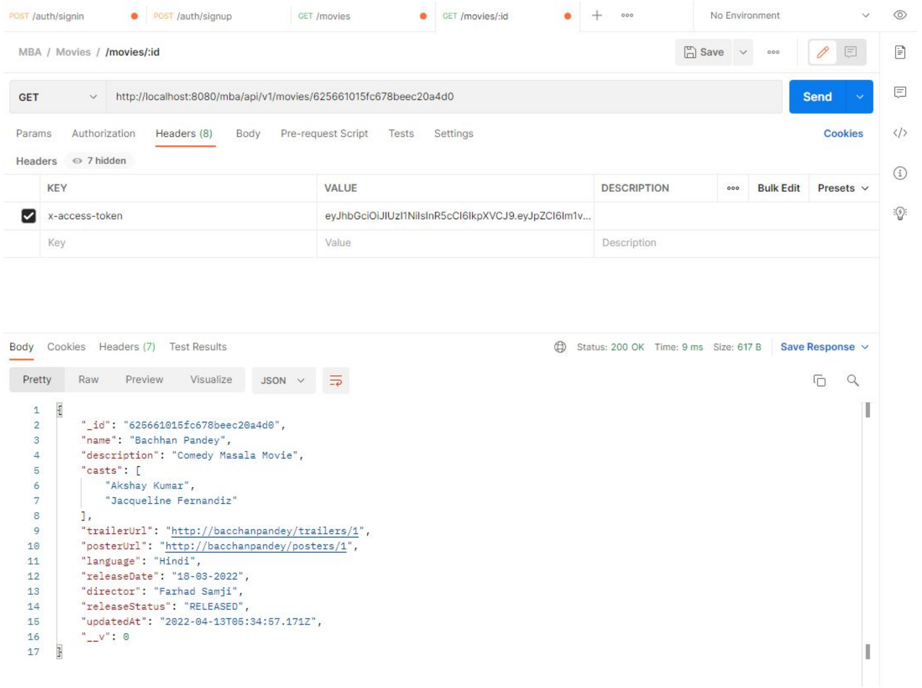
```
{
    "message": "Unauthorized!"
}
```

GET /mba/api/v1/movies/:id

With a token:

- Here, before sending the request we have added a parameter named “x-access-token” with valid jwt token in the request header.
- We have received a successful response here.

GET /mba/api/v1/movies/:id


POST /auth/signin | POST /auth/signup | GET /movies | GET /movies/:id | No Environment

MBA / Movies / /movies/:id

GET http://localhost:8080/mba/api/v1/movies/625661015fc678beec20a4d0

Headers (8) — 7 hidden

KEY	VALUE	DESCRIPTION
x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6Im1v...	

Status: 200 OK Time: 9 ms Size: 617 B

```json
{
    "_id": "625661015fc678beec20a4d0",
    "name": "Bachhan Pandey",
    "description": "Comedy Masala Movie",
    "casts": [
        "Akshay Kumar",
        "Jacqueline Fernandiz"
    ],
    "trailerUrl": "http://bacchanpandey/trailers/1",
    "posterUrl": "http://bacchanpandey/posters/1",
    "language": "Hindi",
    "releaseDate": "18-03-2022",
    "director": "Farhad Samji",
    "releaseStatus": "RELEASED",
    "updatedAt": "2022-04-13T05:34:57.171Z",
    "__v": 0
}
```

Assignment questions

1. Create an endpoint to get the list of all the users.
2. Add a middleware to make this endpoint only accessible to admins.

In the upcoming class:

- Set up data model for booking and transaction
- Authenticated APIs for allowing authenticated customers to perform booking
- Ability to cancel the booking
- Ability to complete payment within a given time frame

Thank You!