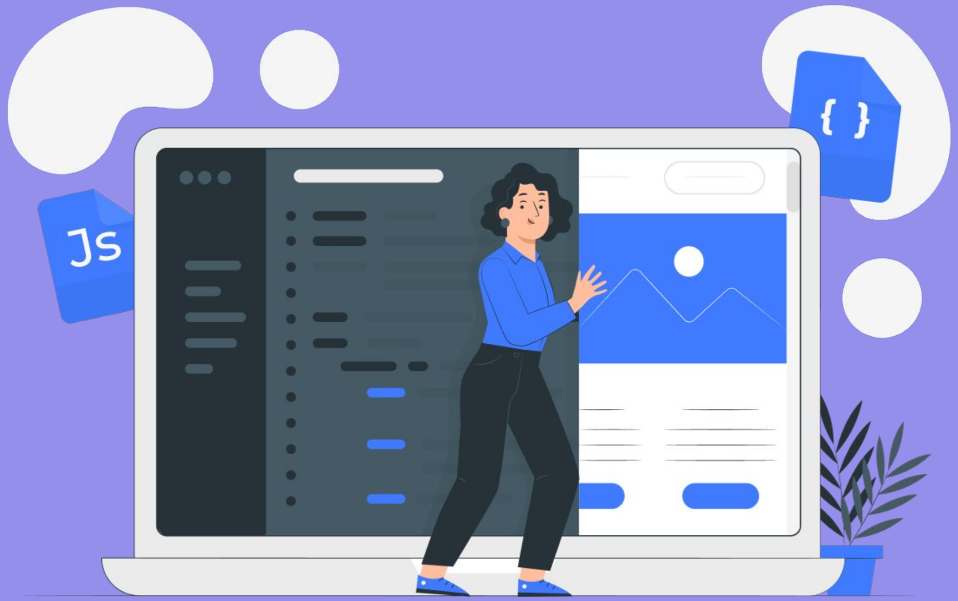
Building the CRM App: Ticket Manipulations and ADMIN capabilities



Relevel
by Unacademy

Topics to be Covered

1. Ticket Manipulations

- RESTful APIs to search tickets by Engineer
- RESTful APIs to update tickets by Engineer
- RESTful APIs to accept tickets by Engineer

2. Review Tickets by Admin

- RESTful APIs to view tickets by Admin
- RESTful APIs to view customers by Admin
- RESTful APIs to view filtered tickets by Admin

Feature of CRM Application to be developed in this session

- API for authenticated Engineer to update the ticket
- The updated ticket should be visible to the customers immediately
- API for authenticated Engineer to search for the ticket
- API for authenticated Engineer to be able to accept a ticket
- API for authenticated Engineer to be able to see the complete list of tickets assigned to him/her
- API for the authenticated ADMIN to get the list of all the customers
- API for the authenticated ADMIN to get the list of all the issues
- API for the authenticated ADMIN to get the list of all the issues after applying certain filters
- API for the authenticated ADMIN to get the list of all the active issues
- API for the authenticated ADMIN to get the list of all the ENGINEER registration requests

- As we have seen in our previous sessions, we are going to use 3 actors which are Customer, Engineer, and Admin.
-
- In this session, we are going to implement below operations of Admin -
- I should be able to see all the tickets details
- I should be able to see all the customers
- I should be able to see all the active tickets
- I should be able to see all the Engineers
- Operations of Engineer
- I should be able to update an issue
- I should be able to search for an issue
- I should be able to accept an issue
- I should be able to see the complete list of issues assigned to me

API – Authenticated Engineer to be able to see the complete list of tickets assigned to him/her

When an authenticated Engineer is trying to view tickets, he should be able to get the complete list of tickets assigned to him
Here, if we pass status as a query parameter, then we will receive tickets only with that status parameter.
We will use userId to fetch the tickets from the database.
Also, if a user is Engineer, we will set assignee as userId of engineer. This will only fetch the tickets who have been assigned to Engineer
URI - GET /crm/api/v1/tickets/

STEP 1 - We will check if the *status* is provided in the request, then we will set status as our query parameter to filter data from the database

```
const queryObj = {};

  if (req.query.status != undefined) {
      queryObj.status = req.query.status;
  }
```

STEP 2 - We will fetch the userId and if the user type is ENGINEER, then we will set assignee as userId of Engineer and add an assignee to the query parameter.

- If the user type is ADMIN, then all the tickets will be returned

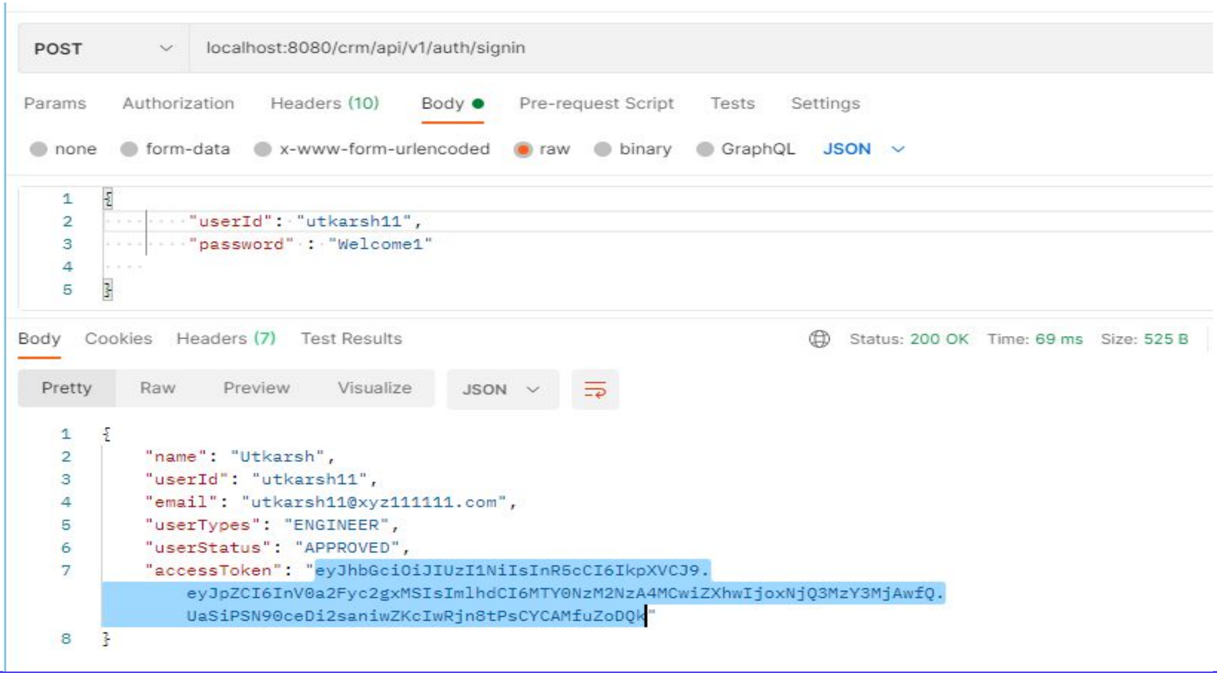
```
     const savedUser = await User.findOne({
            userId: req.userId
        });

        if (savedUser.userType == constants.userTypes.admin) {
            //Do nothing
        } else if (savedUser.userType ==
    constants.userTypes.engineer) {
            queryObj.assignee = req.userId;
        } else {
            queryObj.reporter = req.userId;
        }
```

STEP 3 - Now, we will use query parameters to fetch tickets from the Ticket schema in the database. We will map our response with Ticket model and return.

```
     const tickets = await Ticket.find(queryObj);
     res.status(200).send(objectConvertor.ticketListResponse(ti
ckets));
```

Request

STEP 1 - Login using Engineer userID and copy access token from the response



POST localhost:8080/crm/api/v1/auth/signin

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

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | JSON

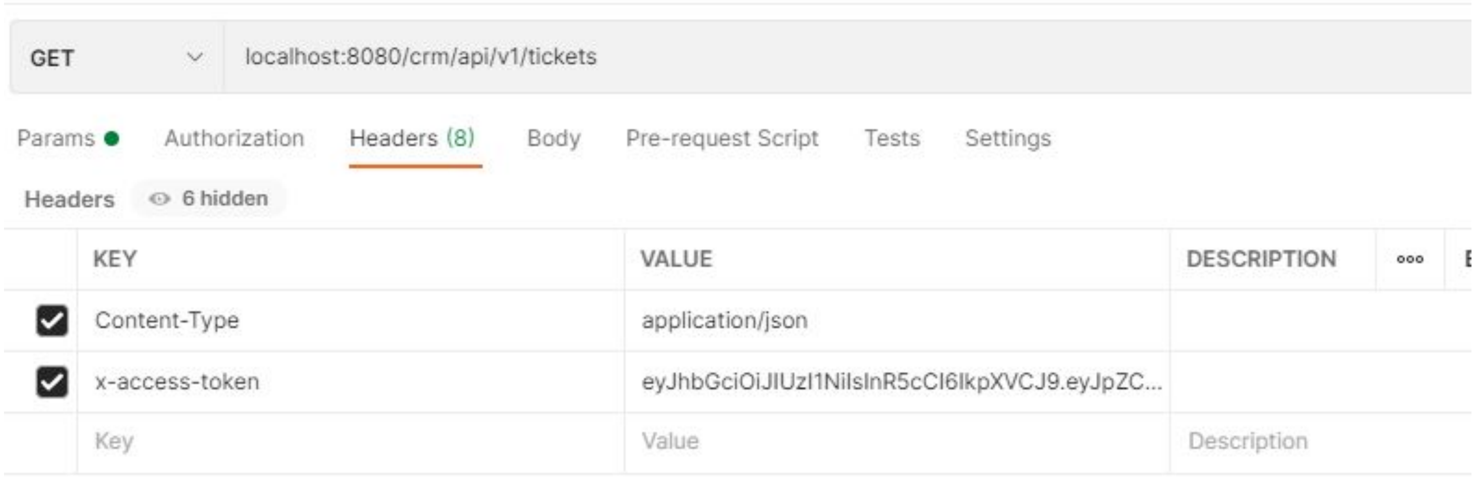
```
{
    "userId": "utkarsh11",
    "password" : "Welcome1"

}
```

Body | Cookies | Headers (7) | Test Results — Status: 200 OK Time: 69 ms Size: 525 B

Pretty | Raw | Preview | Visualize | JSON

```
{
    "name": "Utkarsh",
    "userId": "utkarsh11",
    "email": "utkarsh11@xyz111111.com",
    "userTypes": "ENGINEER",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6InV0a2Fyc2gxMSIsImlhdCI6MTY0NzM2NzA4MCwiZXhwIjoxNjQ3MzY3MjAwfQ.
        UaSiPSN90ceDi2saniwZKcIwRjn8tPsCYCAMfuZoDQk"
}
```

Request - Header

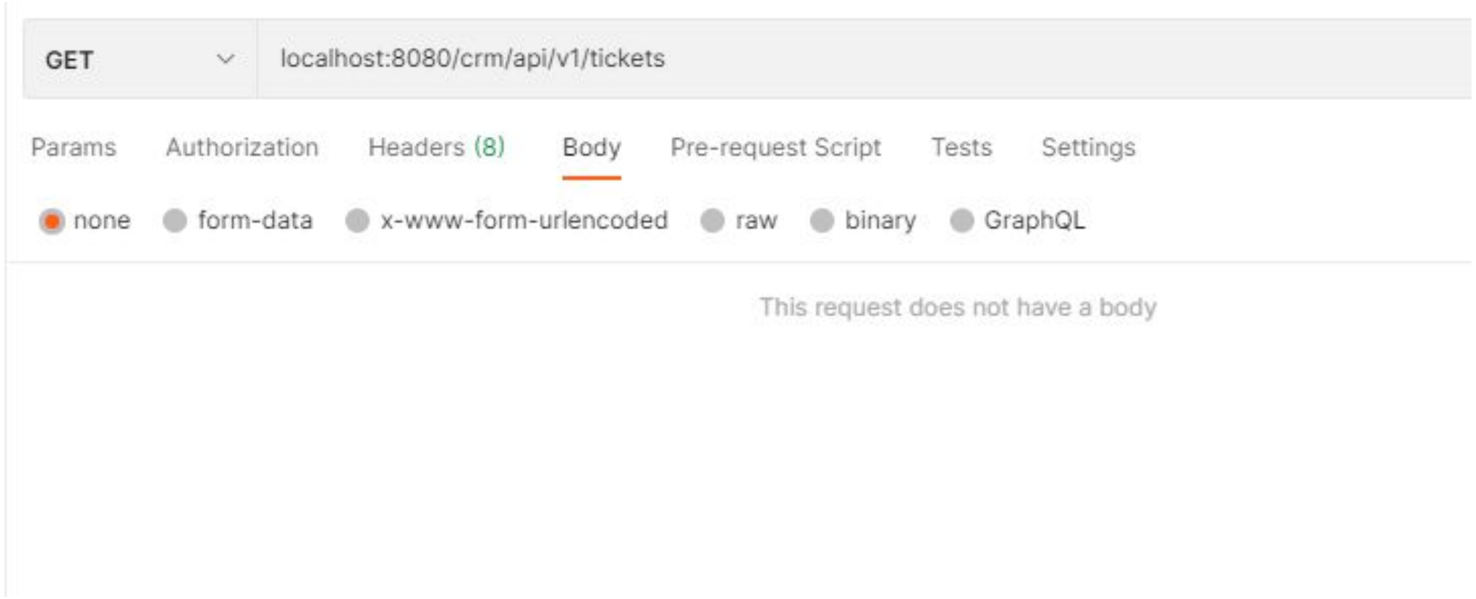

GET localhost:8080/crm/api/v1/tickets

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

Headers 6 hidden

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC...	
	Key	Value	Description

Request - Body

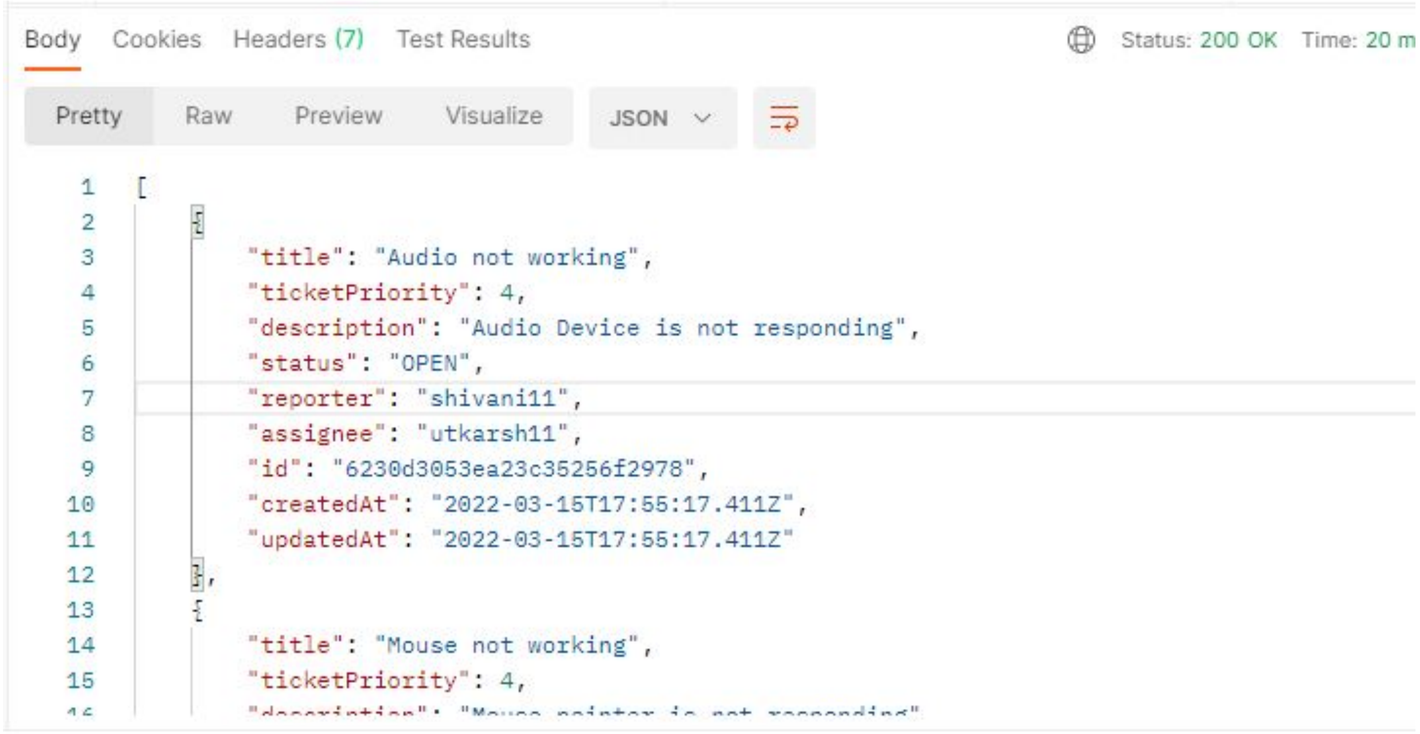

GET localhost:8080/crm/api/v1/tickets

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

[✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Response


Body Cookies Headers (7) Test Results Status: 200 OK Time: 20 m

Pretty Raw Preview Visualize JSON

```
[
    {
        "title": "Audio not working",
        "ticketPriority": 4,
        "description": "Audio Device is not responding",
        "status": "OPEN",
        "reporter": "shivani11",
        "assignee": "utkarsh11",
        "id": "6230d3053ea23c35256f2978",
        "createdAt": "2022-03-15T17:55:17.411Z",
        "updatedAt": "2022-03-15T17:55:17.411Z"
    },
    {
        "title": "Mouse not working",
        "ticketPriority": 4,
```

API – Authenticated Engineer search for the ticket

Our operation is to search for the ticket based on the ticketId by Engineer.

Steps -.

Our operation is to fetch the ticket based on the ticketId given in the request.

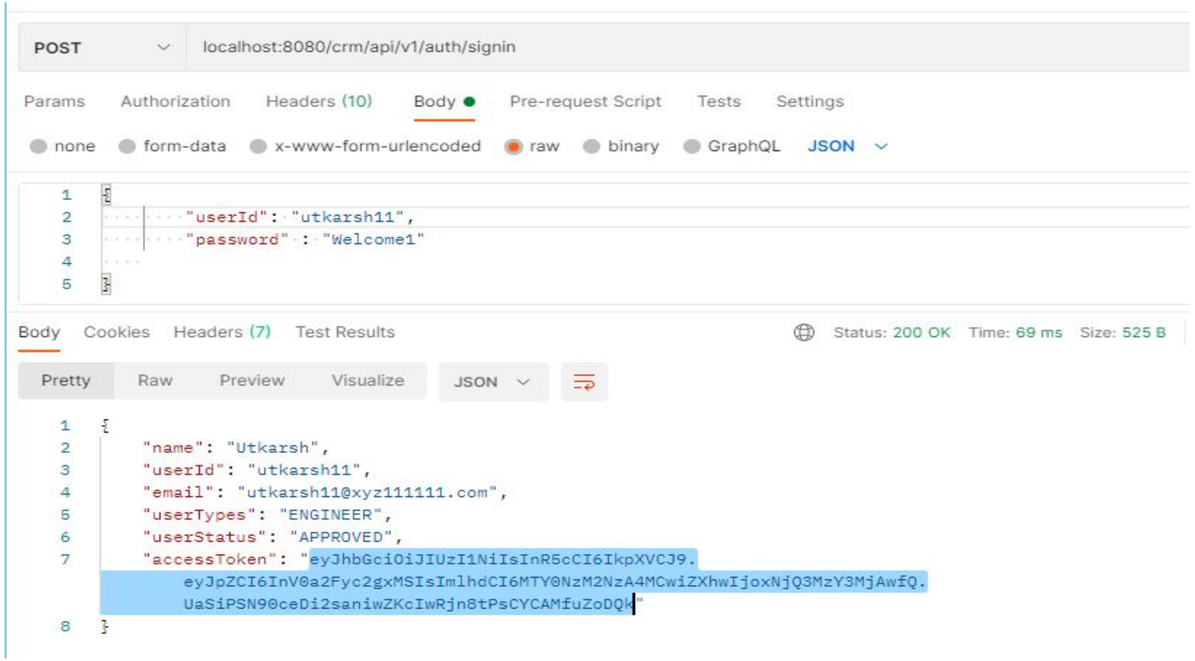
We need to focus on the points below -

- We will use findOne() function of mongoDB
- We will fetch ticket Id from the request and use it to fetch the ticket

```
/**
 * Get the ticket based on the ticketId
 */

exports.getOneTicket = async (req, res) => {
    const ticket = await Ticket.findOne({
        _id: req.params.id
    })

    res.status(200).send(objectConvertor.ticketResponse(ticket));
}
```

Request

STEP 1 - Login using Engineer userID and copy access token from the response



POST localhost:8080/crm/api/v1/auth/signin

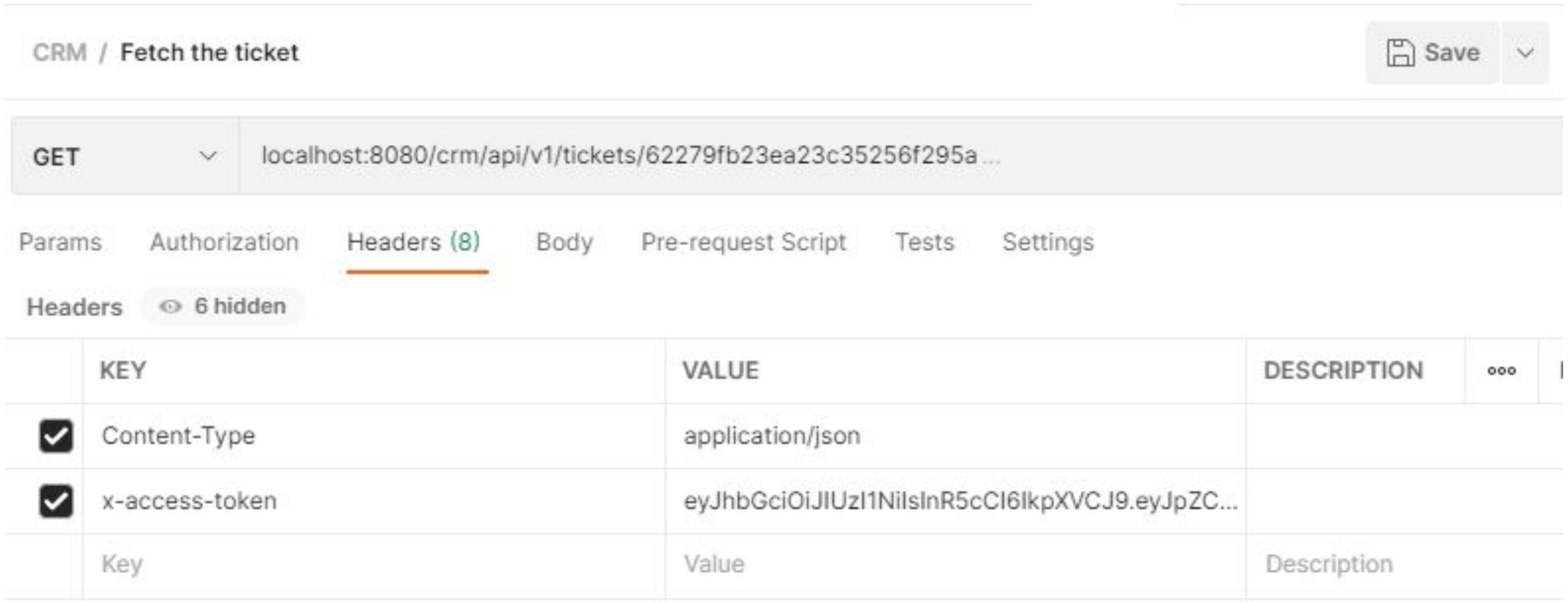
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "userId": "utkarsh11",
    "password" : "Welcome1"

}
```

Status: 200 OK Time: 69 ms Size: 525 B

```
{
    "name": "Utkarsh",
    "userId": "utkarsh11",
    "email": "utkarsh11@xyz111111.com",
    "userTypes": "ENGINEER",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6InV0a2Fyc2gxMSIsImlhdCI6MTY0NzM2NzA4MCwiZXhwIjoxNjQ3MzY3MjAwfQ.
        UaSiPSN90ceDi2saniwZKcIwRjn8tPsCYCAMfuZoDQk"
}
```

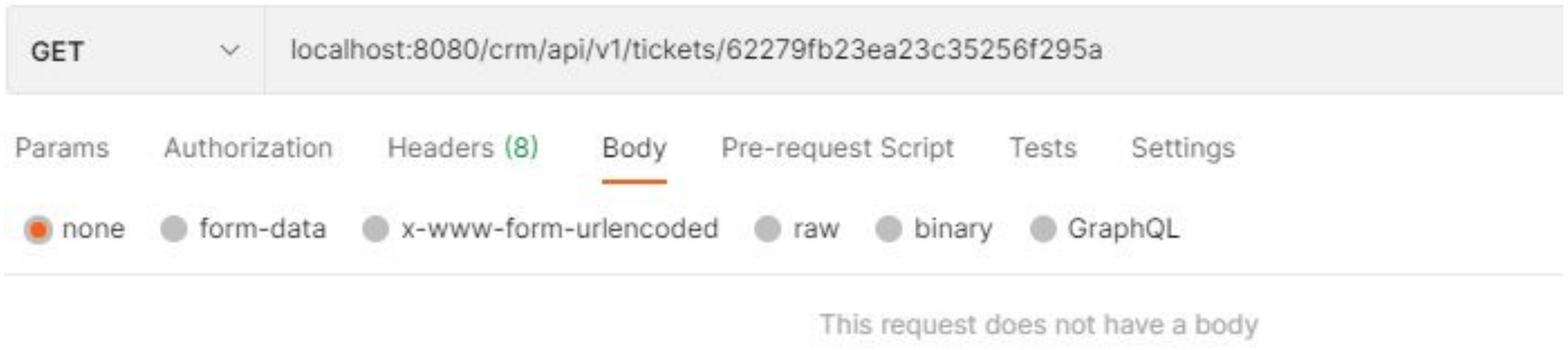
Request - Header



localhost:8080/crm/api/v1/tickets/62279fb23ea23c35256f295a ...

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC...	
	Key	Value	Description

Request - Body

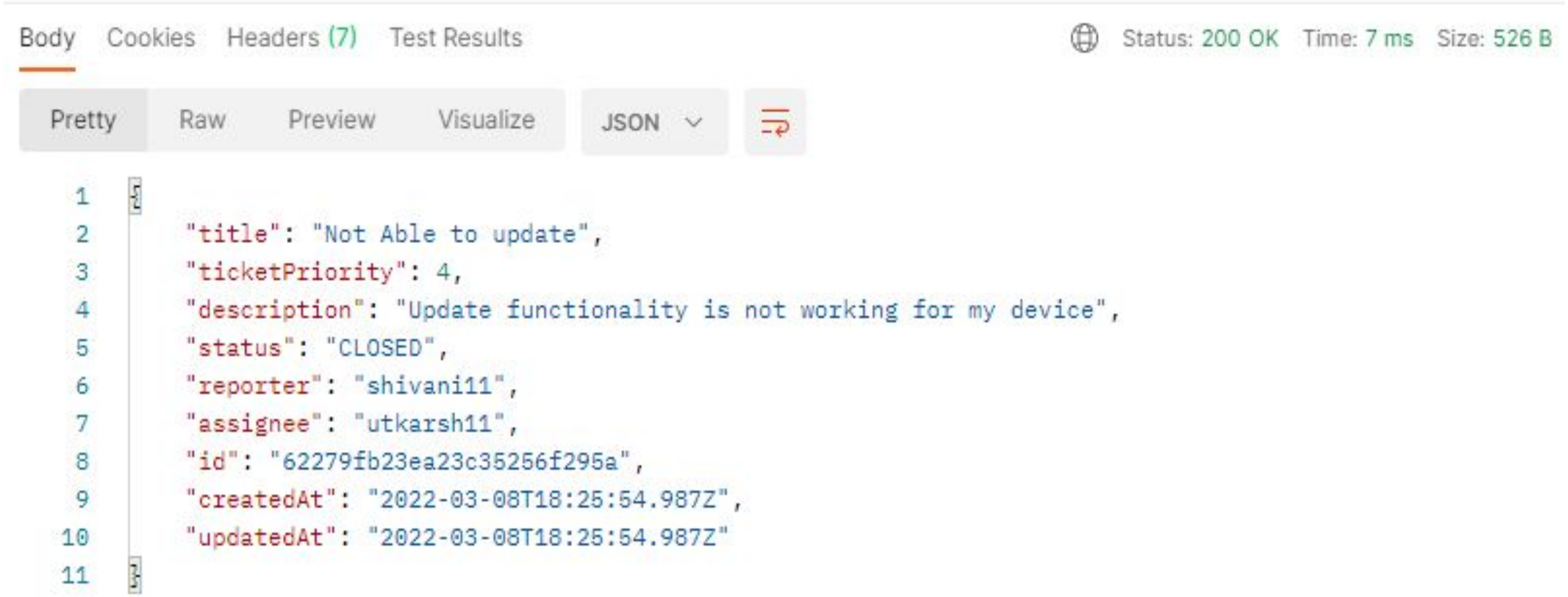


GET localhost:8080/crm/api/v1/tickets/62279fb23ea23c35256f295a

[✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Response



```
{
    "title": "Not Able to update",
    "ticketPriority": 4,
    "description": "Update functionality is not working for my device",
    "status": "CLOSED",
    "reporter": "shivani11",
    "assignee": "utkarsh11",
    "id": "62279fb23ea23c35256f295a",
    "createdAt": "2022-03-08T18:25:54.987Z",
    "updatedAt": "2022-03-08T18:25:54.987Z"
}
```

API – Fetch all the tickets by authenticated ADMIN

Our operation is to fetch all the tickets created by the user.

We need to focus on the points below -

We will use find() function of mongoDB

We will fetch userId and status from the request and use these parameters to fetch the tickets

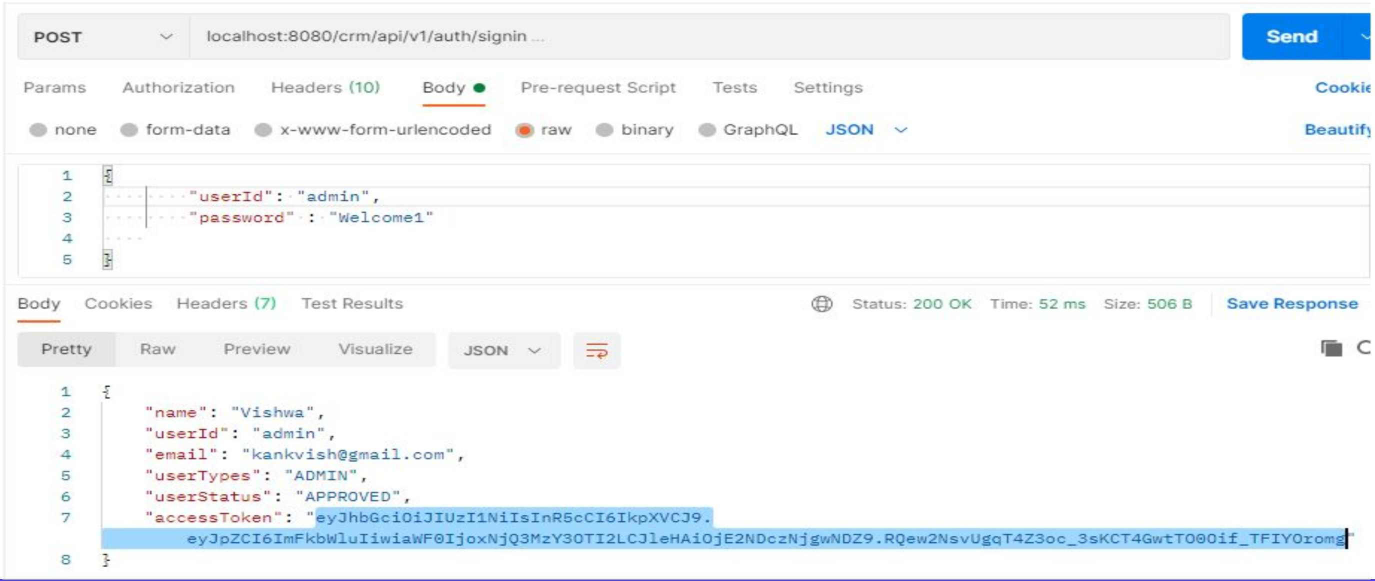
If we are not passing status in the request, then it will fetch all the tickets based on the userId only

```
exports.getAllTickets = async (req, res) => {

    const queryObj = {
        reporter: req.userId
    }

    if (req.query.status != undefined) {
        queryObj.status = req.query.status;
    }


    const tickets = await Ticket.find(queryObj);
    res.status(200).send(objectConvertor.ticketListResponse(tickets));
}
```

Request

Scenario 1 - When we don't pass any filter

STEP 1 - Login using Admin userID and copy access token from the response

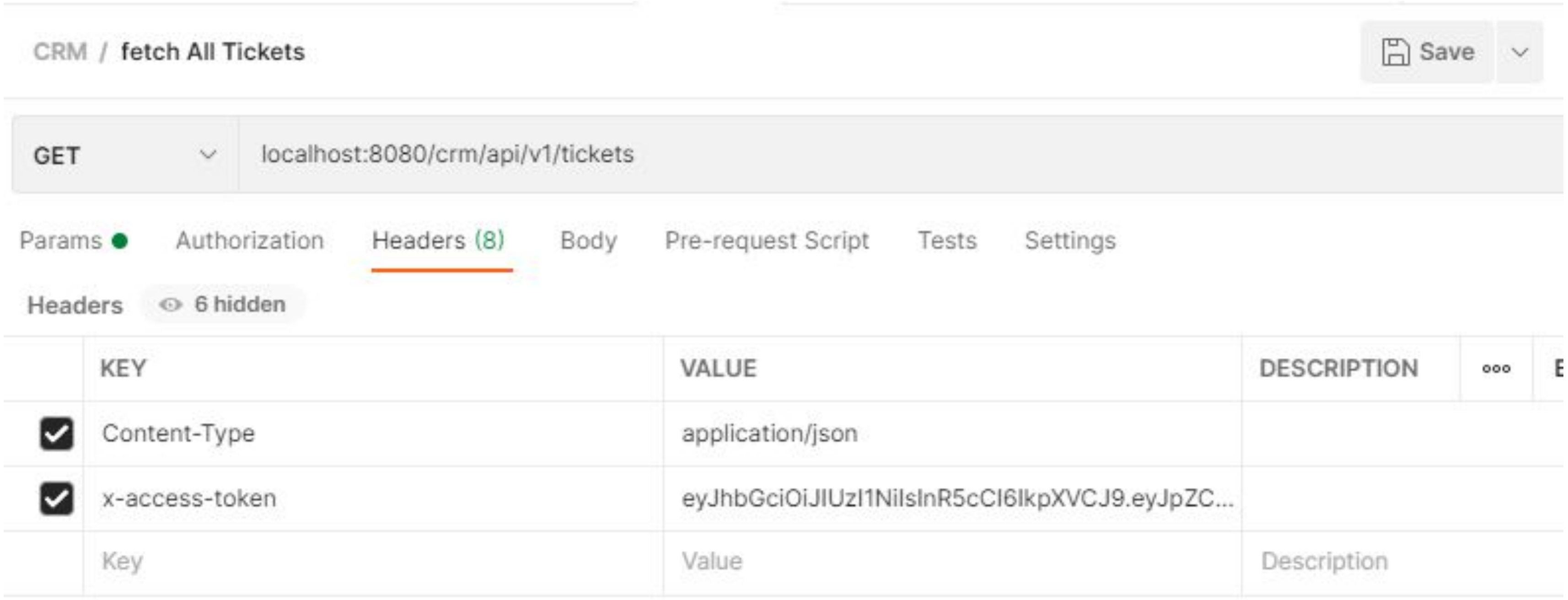

POST localhost:8080/crm/api/v1/auth/signin

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

}
```

Status: 200 OK Time: 52 ms Size: 506 B

```
{
    "name": "Vishwa",
    "userId": "admin",
    "email": "kankvish@gmail.com",
    "userTypes": "ADMIN",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6ImFkbWluIiwiaWF0IjoxNjQ3MzY3OTI2LCJleHAiOjE2NDczNjgwNDZ9.RQew2NsvUgqT4Z3oc_3sKCT4GwtTO0Oif_TFIYOromg"
}
```

Request - Header

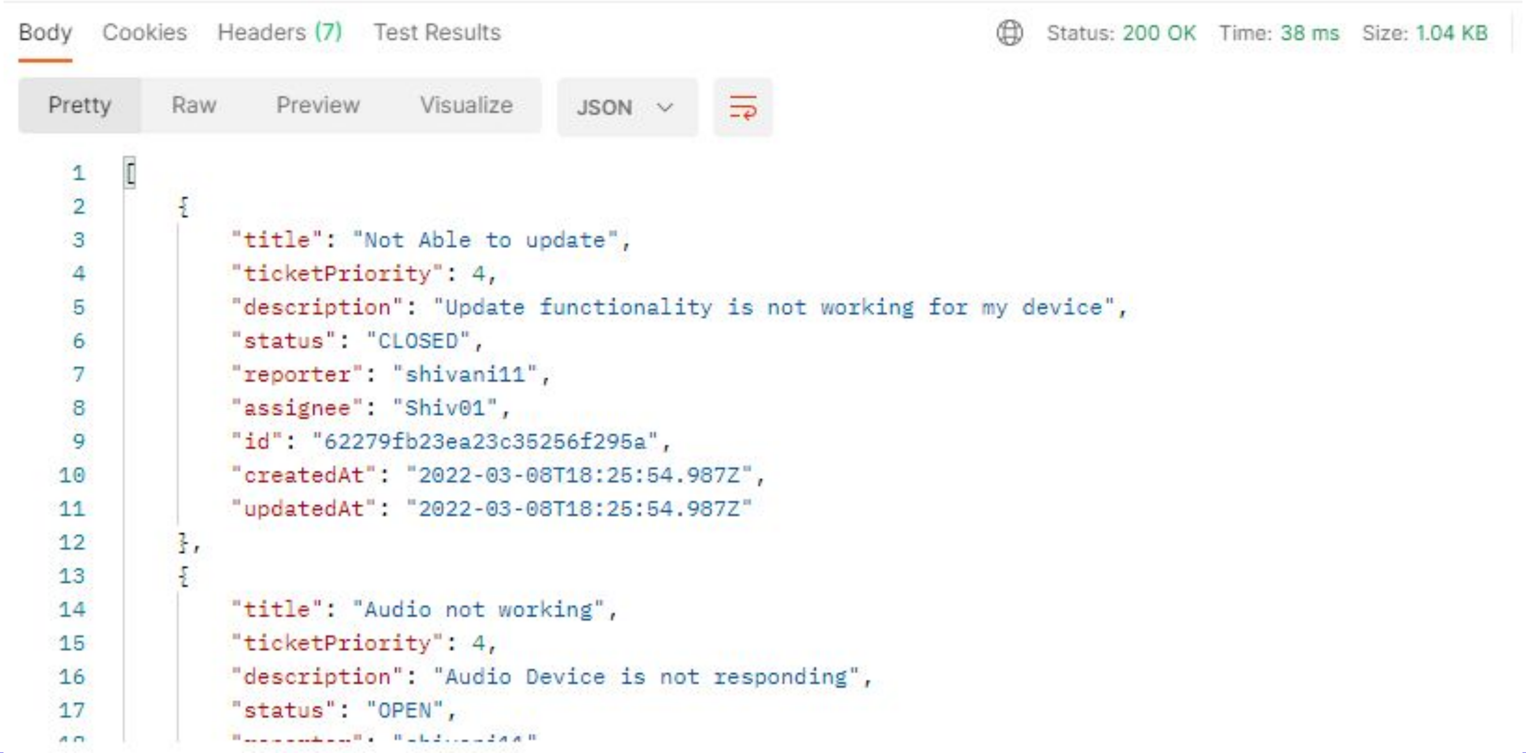


CRM / fetch All Tickets

GET localhost:8080/crm/api/v1/tickets

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC...	
	Key	Value	Description

Response



Body Cookies Headers (7) Test Results

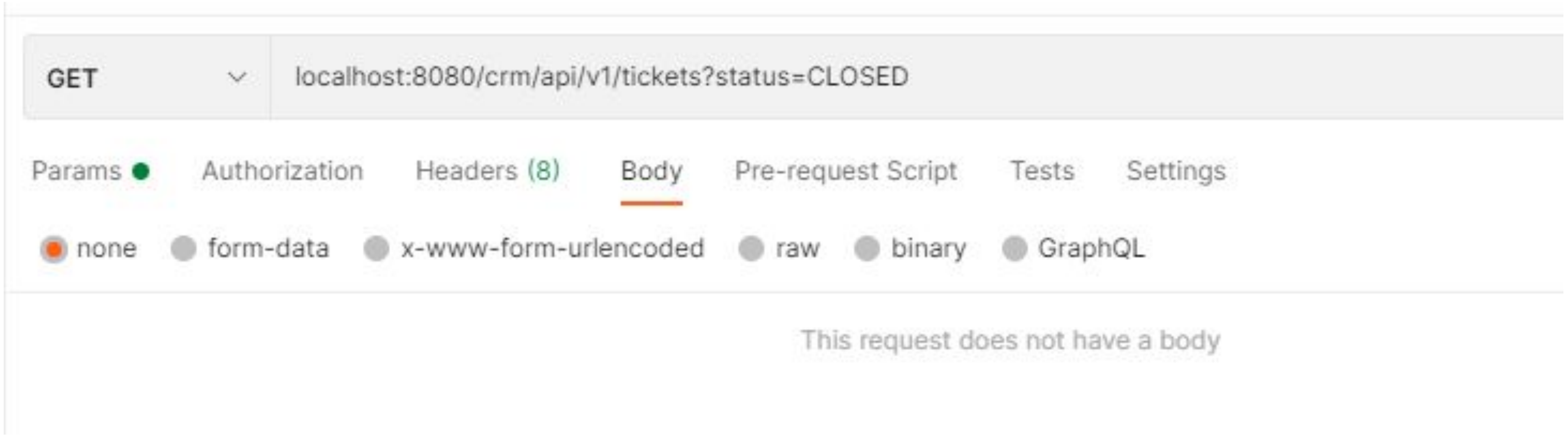
Status: 200 OK Time: 38 ms Size: 1.04 KB

Pretty Raw Preview Visualize JSON

```
[
    {
        "title": "Not Able to update",
        "ticketPriority": 4,
        "description": "Update functionality is not working for my device",
        "status": "CLOSED",
        "reporter": "shivani11",
        "assignee": "Shiv01",
        "id": "62279fb23ea23c35256f295a",
        "createdAt": "2022-03-08T18:25:54.987Z",
        "updatedAt": "2022-03-08T18:25:54.987Z"
    },
    {
        "title": "Audio not working",
        "ticketPriority": 4,
        "description": "Audio Device is not responding",
        "status": "OPEN",
```

Scenario 1 - When we pass status as a filter

Request

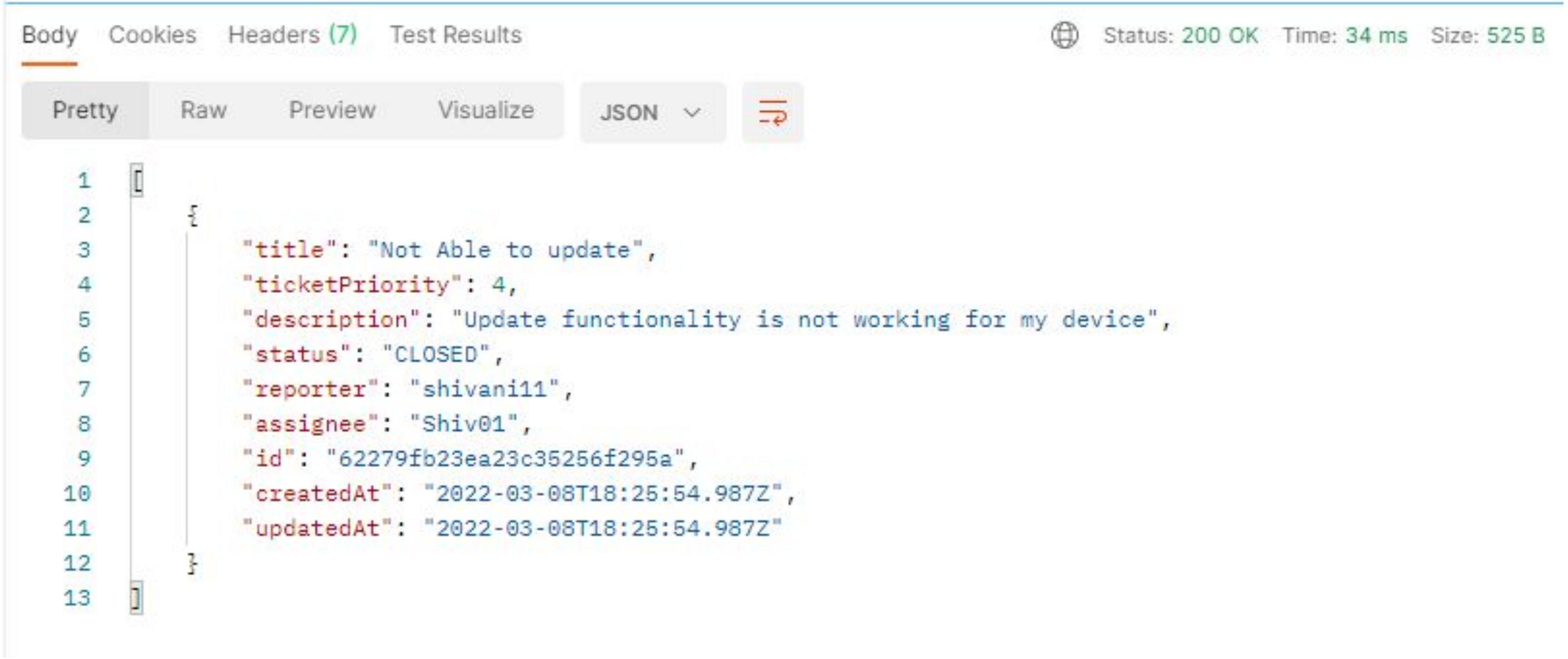

GET localhost:8080/crm/api/v1/tickets?status=CLOSED

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

[✓ none] form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Response



```
[
    {
        "title": "Not Able to update",
        "ticketPriority": 4,
        "description": "Update functionality is not working for my device",
        "status": "CLOSED",
        "reporter": "shivani11",
        "assignee": "Shiv01",
        "id": "62279fb23ea23c35256f295a",
        "createdAt": "2022-03-08T18:25:54.987Z",
        "updatedAt": "2022-03-08T18:25:54.987Z"
    }
]
```

API – Update the Ticket/Reassign the Ticket by Engineer

Our operation is to update the ticket by Engineer. Engineer can reassign the ticket to someone else.

Steps -.

STEP 1 - Before updating a ticket, we need to validate the status of the ticket. We will verify the status from the available status list which are “CLOSED”, “BLOCKED”, “IN_PROGRESS, “OPEN”

```
validateTicketStatus = async (req, res, next) => {
    //Validating the user type
    const status = req.body.status;
    const statusTypes = [constants.ticketStatus.open,
constants.ticketStatus.closed, constants.ticketStatus.inProgress,
constants.ticketStatus.blocked]
    if (status && !statusTypes.includes(status)) {
        res.status(400).send({
            message: "status provided is invalid. Possible values CLOSED
| BLOCKED | IN_PROGESS | OPEN "
        });
        return;
    }

    next();
}
```

STEP 2 - Now, we will call our actual function to update the ticket.
We need to focus on the points below before updating the ticket.

- Only the user who has created the ticket is allowed to update the ticket
- Engineer who is assignee of the ticket is allowed to update the ticket
- Admin is allowed to update the ticket
- In our case, the Engineer is updating the ticket, so we will check the assignee of the ticket and compare it with userId and if both are same, then Engineer is allowed to update the ticket
- We will fetch all the information from the request and update the ticket in the database using *save()* function.

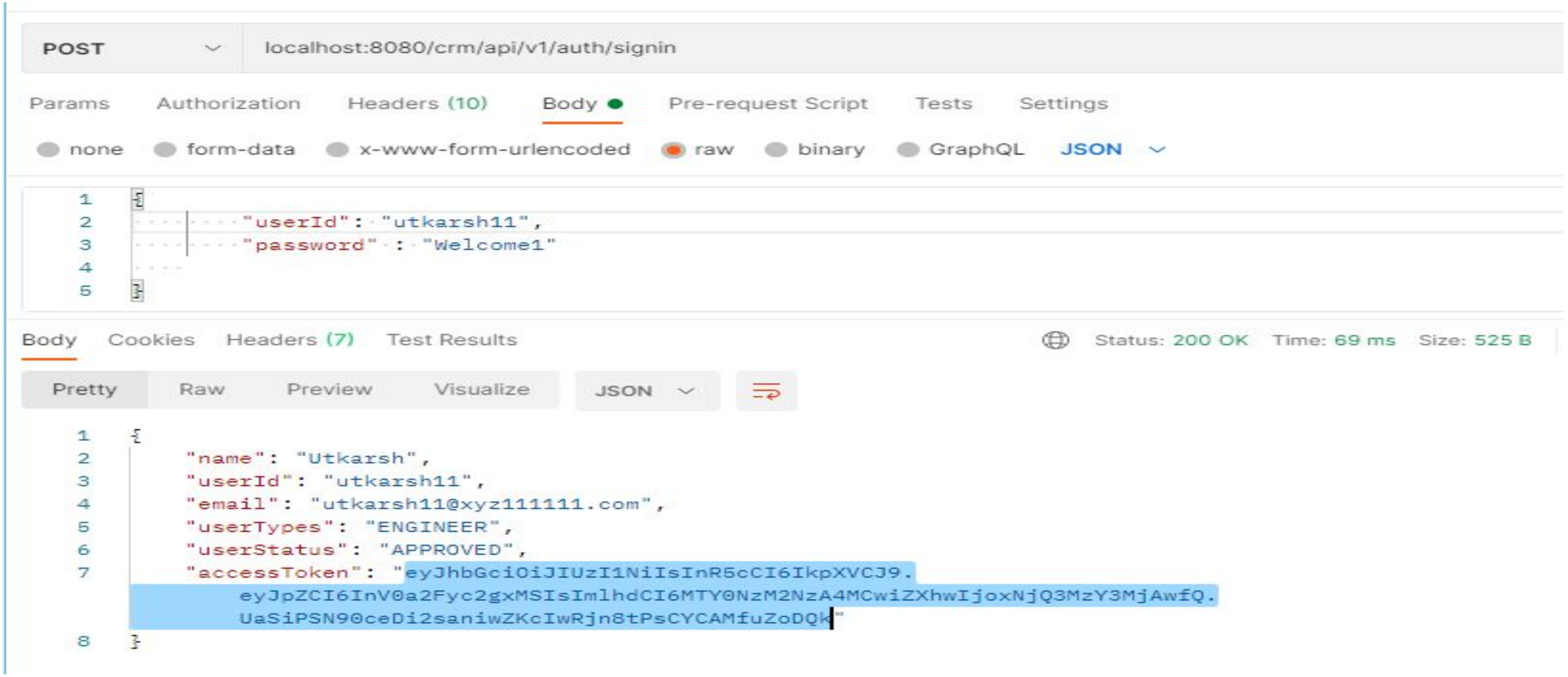
```
const ticket = await Ticket.findOne({ _id: req.params.id });
     const savedUser = await User.findOne({
        userId: req.userId
    });
     if (ticket.reporter == req.userId || ticket.assignee == req.userId ||
savedUser.userType == constants.userTypes.admin) {
        //Allowed to update
        ticket.title = req.body.title != undefined ? req.body.title :
ticket.title,
            ticket.description = req.body.description != undefined ?
req.body.description : ticket.description,
            ticket.ticketPriority = req.body.ticketPriority != undefined ?
req.body.ticketPriority : ticket.ticketPriority,
            ticket.status = req.body.status != undefined ? req.body.status
: ticket.status,
            ticket.assignee = req.body.assignee !=undefined ?
req.body.assignee : ticket.assignee
```

```
            var updatedTicket = await ticket.save();
           res.status(200).send(objectConvertor.ticketResponse(updatedTicket)
);
     } else {
           console.log("Ticket was being updated by someone who has not
created the ticket");
           res.status(401).send({
               message: "Ticket can be updated only by the customer who
created it"
           })
```

Request

Scenario 1 - When we don't pass any filter

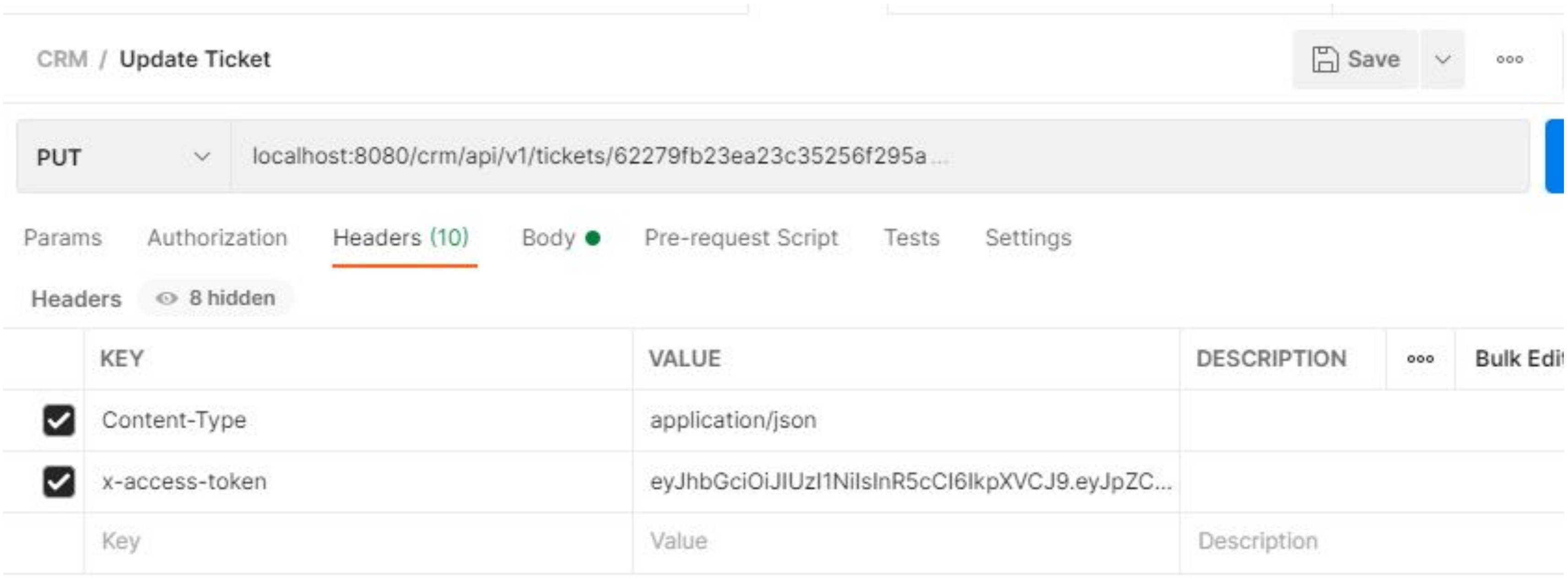
STEP 1 - Login using Engineer userID and copy access token from the response



```
{
    "userId": "utkarsh11",
    "password" : "Welcome1"

}
```

```
{
    "name": "Utkarsh",
    "userId": "utkarsh11",
    "email": "utkarsh11@xyz111111.com",
    "userTypes": "ENGINEER",
    "userStatus": "APPROVED",
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJpZCI6InV0a2Fyc2gxMSIsImlhdCI6MTY0NzM2NzA4MCwiZXhwIjoxNjQ3MzY3MjAwfQ.
        UaSiPSN90ceDi2saniwZKcIwRjn8tPsCYCAMfuZoDQk"
}
```

Request - Header


CRM / Update Ticket

Save

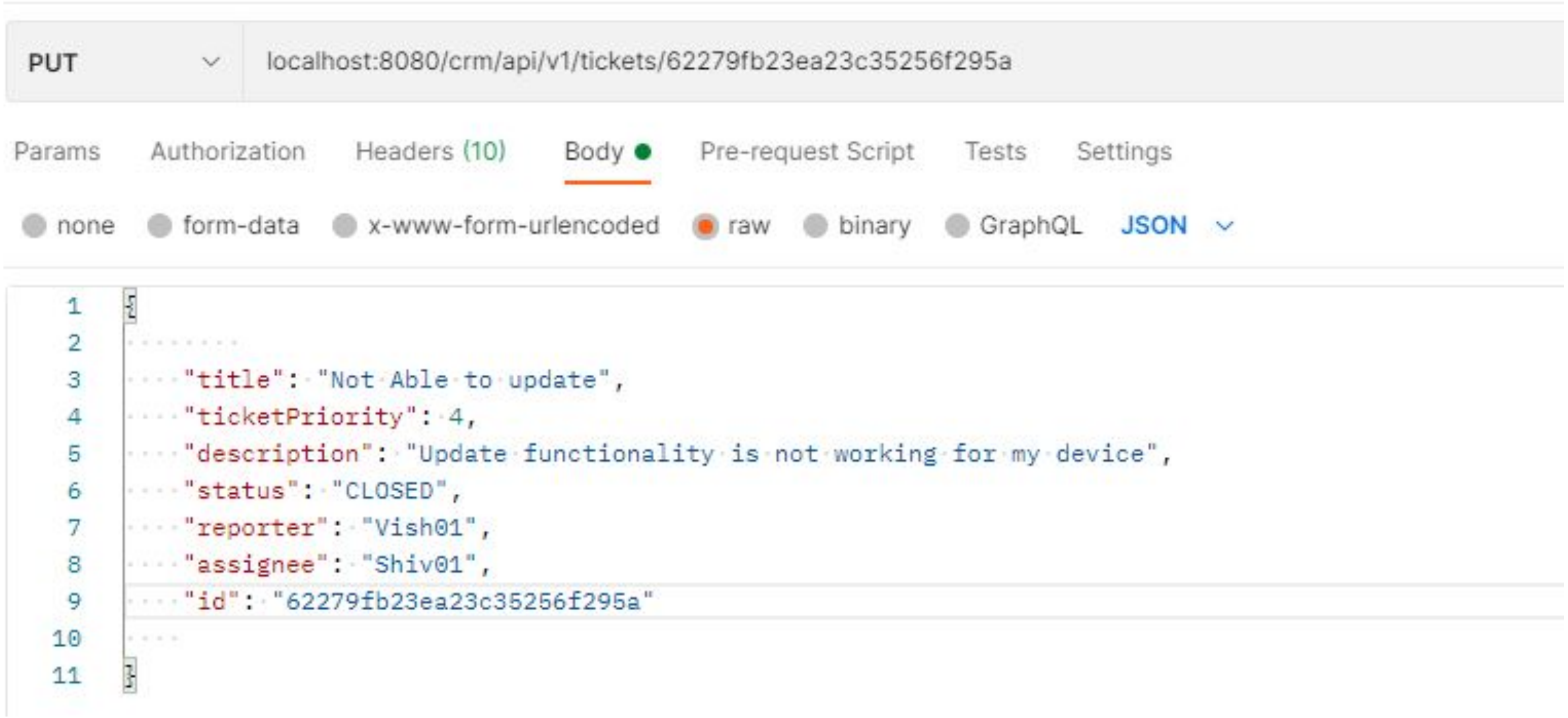
PUT localhost:8080/crm/api/v1/tickets/62279fb23ea23c35256f295a ...

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

Headers 8 hidden

	KEY	VALUE	DESCRIPTION
✓	Content-Type	application/json	
✓	x-access-token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC...	
	Key	Value	Description

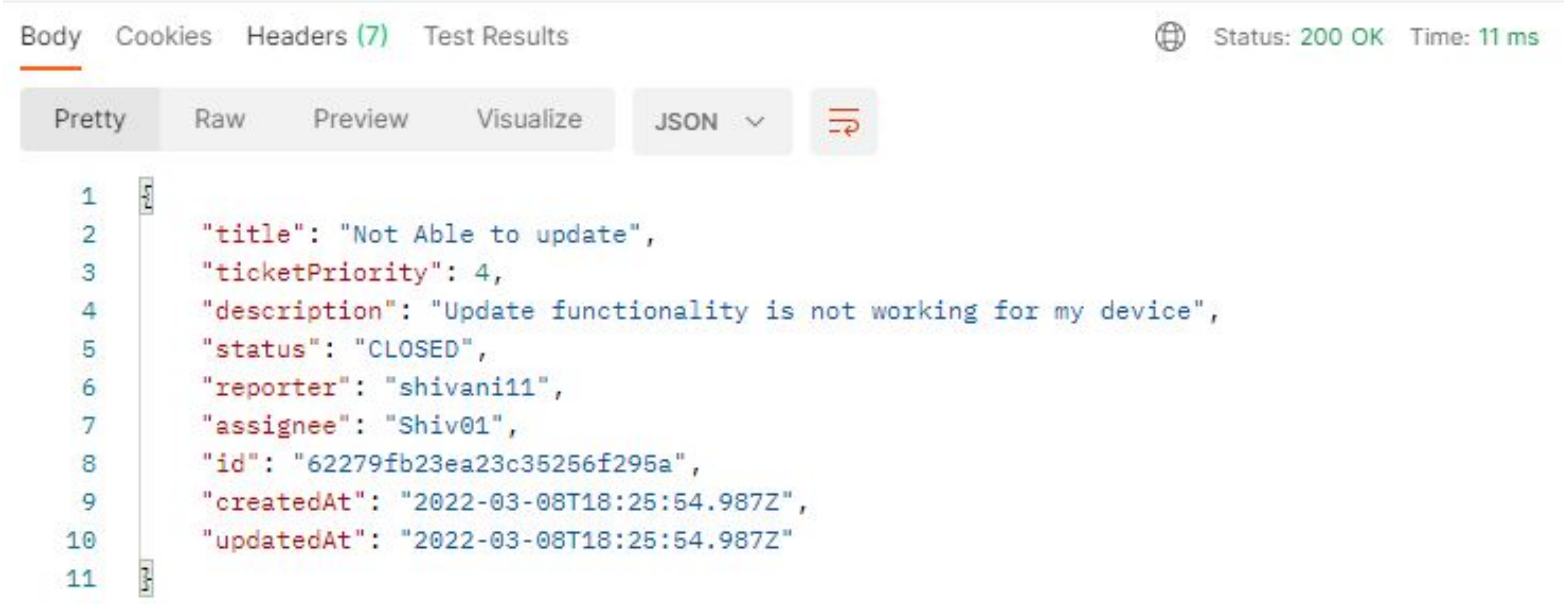
Request



localhost:8080/crm/api/v1/tickets/62279fb23ea23c35256f295a

```
{

    "title": "Not Able to update",
    "ticketPriority": 4,
    "description": "Update functionality is not working for my device",
    "status": "CLOSED",
    "reporter": "Vish01",
    "assignee": "Shiv01",
    "id": "62279fb23ea23c35256f295a"
    
}
```

Response



```
{
    "title": "Not Able to update",
    "ticketPriority": 4,
    "description": "Update functionality is not working for my device",
    "status": "CLOSED",
    "reporter": "shivani11",
    "assignee": "Shiv01",
    "id": "62279fb23ea23c35256f295a",
    "createdAt": "2022-03-08T18:25:54.987Z",
    "updatedAt": "2022-03-08T18:25:54.987Z"
}
```

Practice Code

- Write an API to fetch all the customers by an admin user
- Write an API to fetch all the tickets raised by the customer

MCQ Questions

1. **What assignee is referring to in the Ticket schema?**

A. Admin
B. Customer
C. Engineer
D. None

Ans: C

2. **What reporter is referring to in the Ticket schema?**

A. Admin
B. Customer
C. Engineer
D. None

Ans: B

3. **What is the below URL performing in our code when requested by the Customer?**

GET /crm/api/v1/tickets/

A. Fetching all the tickets raised by the Customer
B. Fetching all the tickets assigned to Engineer
C. A and B
D. None

Ans: A

4. **What is the below URL performing in our code when requested by the Engineer?**

GET /crm/api/v1/tickets/

A. Fetching all the tickets assigned to Engineer
B. Fetching all the tickets raised by Customer
C. A and B
D. None

Ans: A

5. What will be the status of the ticket when completed?

A. CLOSED
B. OPEN
C. IN_PROGRESS
D. BLOCKED

Ans: A

6. Which statement is correct about the PUT method?

A. PUT requests are idempotent
B. When we call PUT requests multiple times, it will return the same output
C. PUT is used to update the resource
D. All of the above

Ans: D

7. Which statement is correct about the POST method?

A. POST requests are not idempotent

B. When we call POST requests multiple times, it may or may not return the same output

C. POST is used to create the whole entity

D. All of the above

Ans: D

THANK YOU