Ganpat University | Institute of Computer Technology

॥ विद्यया समाजोत्कर्षः ॥

Name: Tushar Panchal

En.No: 21162101014

Sub: MICROSEVICES

Branch: CBA

Batch:51

----------PRACTICAL 09----------

❖ Question (TASK) :

Demonstrate the
secured HTTP through SSL.

Symbiosis Pvt Ltd makes NodeJS-based websites and deploys as well. At the time of deployment, websites require some authentication, and encryption policy for security purposes. Apply the following technique and secure said company's website while deploying:

1. ***Generate a Public certificate with a public key and certificate***
2. ***Connect it with the website using the HTTPS library***
3. ***Secure application using Username and password.***
4. ***Secure your REST APIs using the Bearer token technique.***
5. ***Secure your RESTAPIs created in previous practicals using username and password***

Github Link :
https://github.com/Tushar007079/MICROSERVICES_PRACTICALS/tree/1f0806cc7f3ddd3233b671843ae4bf9c02dd1912/9

❖ STEPS TO PERFORM THIS TASK :

1. Generate a Public certificate with a public key and certificate:

- To create a self-signed certificate using OpenSSL: :
  `openssl req -newkey rsa:2048 -nodes -keyout server.key -x509 -days 365 -out server.crt`
- This command generates a self-signed certificate (**server.crt**) and a private key (**server.key**) valid for 365 days.

```
pwsh  9  37ms
>> openssl req -newkey rsa:2048 -nodes -keyout server.key -x509 -days 365 -out server.crt
......+..+....+.....+....+..+....+........+.++++++++++++++++++++++++++++++++++++++++++*..+.+..+....
..+..+............+.........+.++++++++++++++++++++++++++++++++++++++++++*..+..+...+.......+...+....
+......................+.+...+................+....+...............+......+..+......+....+..+...
...+..+...+.+..+.......+..+....+......+...+...+.........+.....+.+.....+......+...+.+........
........+...+.....+...+.+...+...+........+.+.....+.+...............+.+.....+....+.....+.+.....+..
.....+.+........+.+..+....+...+................+.+........++++++++++++++++++++++++++++++++++++
+++*...+.......+.........+.....+......+...............+..................+......+.+...+.....+...
..+...+.+...+...........+.+.....+.+........+.......+..+.......+......+.........+...+......+.....
..+.+............+...+.....+......+.........+....+..+.........+....+..+..........+.........+..+.
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:IN
State or Province Name (full name) [Some-State]:GUJARAT
Locality Name (eg, city) []:AHMEDABAD
Organization Name (eg, company) [Internet Widgits Pty Ltd]:CBA
Organizational Unit Name (eg, section) []:ICT
Common Name (e.g. server FQDN or YOUR name) []:JAMES
                                                BOND
Email Address []:tusharpanchal21@gnu.ac.in
```

2. Connect it with the website using the HTTPS library :

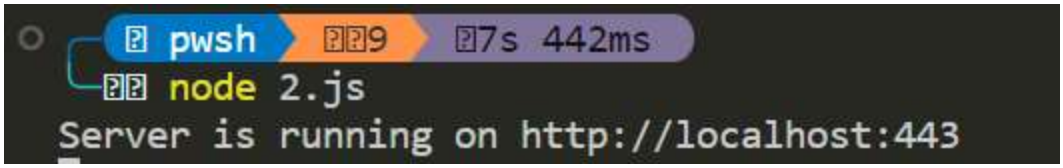
```
const https = require('https');
const fs = require('fs');
const express = require('express');

const app = express();
const port = 443;
```

```
const options = {
  key: fs.readFileSync('server.key'),
  cert: fs.readFileSync('server.crt')
};

const server = https.createServer(options, app);

app.get('/', (req, res) => {
  res.send('Welcome to the Secure Website!');
});

server.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

- Run this following command to initialize the package.json file
  `npm init -y`
- Now, install the required Node.js modules (express) :
  `npm install express`
- Run the server.
- In the Terminal/CMD , run this following command to start the NodeJS Server:
  `node 2.js`

Once you run the code, you should see a message in the console indicating that the server is running on **https://localhost:443**

✓ **Output :**



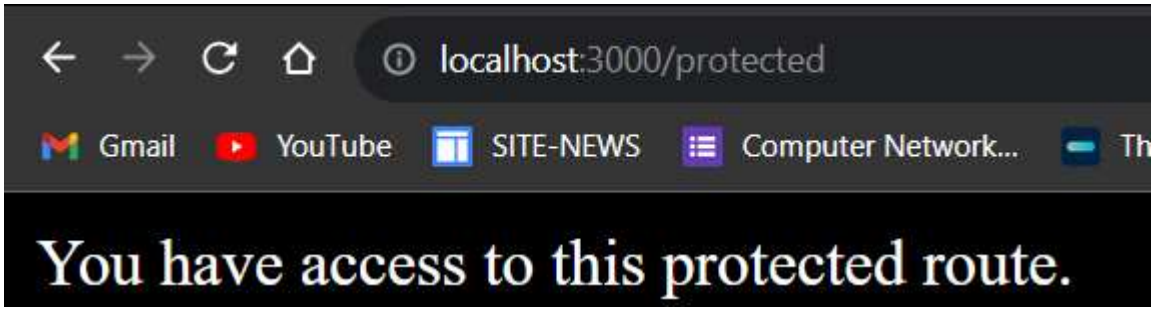
3. Secure application using Username and password :

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const port = 3000;

// Middleware for parsing request body
app.use(bodyParser.urlencoded({ extended: false }));

// In-memory database for storing user credentials (Replace with a database in
a production environment)
const users = [
  { username: 'admin', password: 'admin' },
  { username: 'James Bond', password: '007' },
];

// Middleware for basic authentication
function authentication(req, res, next) {
  const { username, password } = req.body;

  if (!username || !password) {
    const err = new Error('Username and password are required.');
    err.status = 400;
    return next(err);
  }

  const user = users.find((u) => u.username === username && u.password ===
password);

  if (user) {
    next(); // Authorized user
  } else {
    const err = new Error('Invalid username or password.');
    err.status = 401;
    return next(err);
  }
}

// Serve a login form
app.get('/login', (req, res) => {
  res.send(`
    <form method="post" action="/protected">
      <label for="username">Username:</label>
      <input type="text" name="username" id="username"><br>
      <label for="password">Password:</label>
```

```
      <input type="password" name="password" id="password"><br>
      <button type="submit">Login</button>
    </form>
  `);
});

// Apply authentication middleware to protect the protected route
app.post('/protected', authentication, (req, res) => {
  res.send('You have access to this protected route.');
});

// Unprotected route for testing
app.get('/public', (req, res) => {
  res.send('This route is public and doesn\'t require authentication.');
});

// Start the server
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

- you need to install the required Node.js modules. Open your terminal and run the following command :
  `npm install express body-parser`
- In the Terminal/CMD , run this following command to run the NodeJS Server :
  `node 3.js`
  - ✓ **Output :**

Access the login form : **http://localhost:3000/login**



Access the public route : **http://localhost:3000/public/**

4. Secure your REST APIs using the Bearer token technique :

```
const express = require('express');
const jwt = require('jsonwebtoken');

const app = express();
app.use(express.json());

const secretKey = process.env.SECRET_KEY || 'default-secret'; // 'default-
secret' is a fallback in case the environment variable is not set


// Mock user for demonstration
const mockUser = {
  id: Date.now(),
  userEmail: 'tusharpanchal21@gnu.ac.in',
  password: '007',
};

// Route for user login and token generation
app.post('/api/login', (req, res) => {

  // Generate a JWT token
  jwt.sign({ user: mockUser }, secretKey, { expiresIn: '120s' }, (err, token)
=> {
    if (err) {
      res.status(500).json({ error: 'Failed to create token' });
    } else {
```

```
      res.json({ token });
    }
  });
});

// Protected route that requires a valid JWT token
app.get('/api/profile', verifyToken, (req, res) => {
  // The verifyToken middleware verifies the JWT token
  jwt.verify(req.token, secretKey, (err, authData) => {
    if (err) {
      res.sendStatus(403); // Forbidden if token is invalid or expired
    } else {
      res.json({
        message: 'Welcome to Profile',
        userData: authData.user,
      });
    }
  });
});

// Middleware to extract and verify the JWT token from the Authorization
header
function verifyToken(req, res, next) {
  const bearerHeader = req.headers['authorization'];

  if (typeof bearerHeader !== 'undefined') {
    const bearer = bearerHeader.split(' ');
    const bearerToken = bearer[1];

    req.token = bearerToken;
    next();
  } else {
    res.sendStatus(403); // Forbidden if no token is provided in the header
  }
}

const port = 5000;
app.listen(port, () => {
  console.log(`Server started on http://localhost:${port}`);
});
```

- Install the required Node.js modules, which are express and jsonwebtoken. Open your terminal and run the following command: :

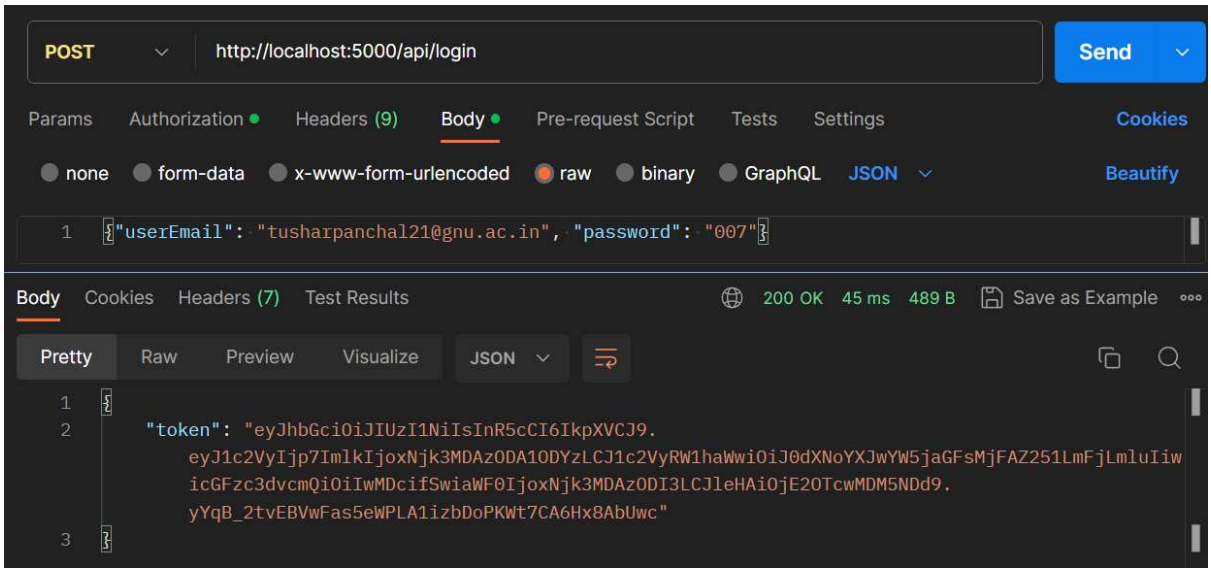
```
npm install express jsonwebtoken
```

» In the Terminal/CMD , run this following command to run the NodeJS Server :

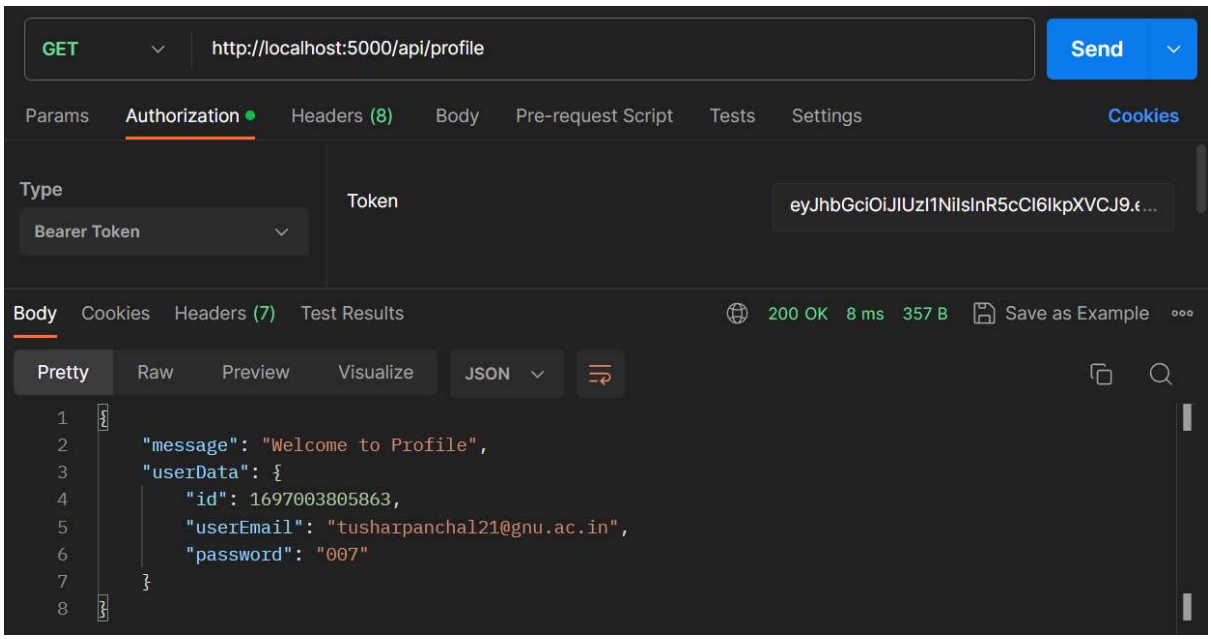
```
node 4.js
```

✓ **Output :**

→ Make a POST Request to /api/login: **http://localhost:5000/api/login**



→ Make a GET Request to /api/profile .In the Headers section, add an Authorization header with the value Bearer YOUR_TOKEN. Replace YOUR_TOKEN with the JWT token you received in the previous step : **http://localhost:5000/api/profile**



5. Secure your RESTAPIs created in previous practicals using username and password :

```
const express = require('express');
const basicAuth = require('basic-auth');

const app = express();
app.use(express.json());

// Dummy user data (replace with your actual user data or database)
const users = [
  { username: 'admin', password: 'password' },
  { username: 'user', password: 'pass123' },
];

// Dummy item data (replace with your actual data or database)
const items = [
  { id: 1, name: 'Item 1' },
  { id: 2, name: 'Item 2' },
];

// Middleware to handle basic authentication
const authenticate = (req, res, next) => {
  const credentials = basicAuth(req);

  if (!credentials) {
    return res.status(401).send('Unauthorized');
  }

  const user = users.find((u) => u.username === credentials.name && u.password
=== credentials.pass);

  if (!user) {
    return res.status(401).send('Unauthorized');
  }

  // If authentication is successful, store the user information in the
request object
  req.user = user;
  next();
};

// Protected API routes
app.get('/api/items', authenticate, (req, res) => {
  // Only authenticated users can access this route
  res.json({ items });
});
```

```
app.get('/api/items/:id', authenticate, (req, res) => {
  const itemId = parseInt(req.params.id);
  const item = items.find((i) => i.id === itemId);

  if (!item) {
    return res.status(404).json({ error: 'Item not found' });
  }

  res.json({ item });
});

app.post('/api/items', authenticate, (req, res) => {
  // Only authenticated users can create items
  const newItem = req.body;
  newItem.id = items.length + 1;
  items.push(newItem);

  res.status(201).json({ message: 'Item created', newItem });
});

app.put('/api/items/:id', authenticate, (req, res) => {
  // Only authenticated users can update items
  const itemId = parseInt(req.params.id);
  const itemIndex = items.findIndex((i) => i.id === itemId);

  if (itemIndex === -1) {
    return res.status(404).json({ error: 'Item not found' });
  }

  const updatedItem = req.body;
  updatedItem.id = itemId;
  items[itemIndex] = updatedItem;

  res.json({ message: 'Item updated', updatedItem });
});

app.delete('/api/items/:id', authenticate, (req, res) => {
  // Only authenticated users can delete items
  const itemId = parseInt(req.params.id);
  const itemIndex = items.findIndex((i) => i.id === itemId);

  if (itemIndex === -1) {
    return res.status(404).json({ error: 'Item not found' });
  }

  const deletedItem = items.splice(itemIndex, 1)[0];
```

```
  res.json({ message: 'Item deleted', deletedItem });
});

// Start the server
const port = process.env.PORT || 3000;
app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

- You need to install the required dependencies, which are express and basic-auth. Open your terminal and run the following command :

  `npm install express basic-auth`

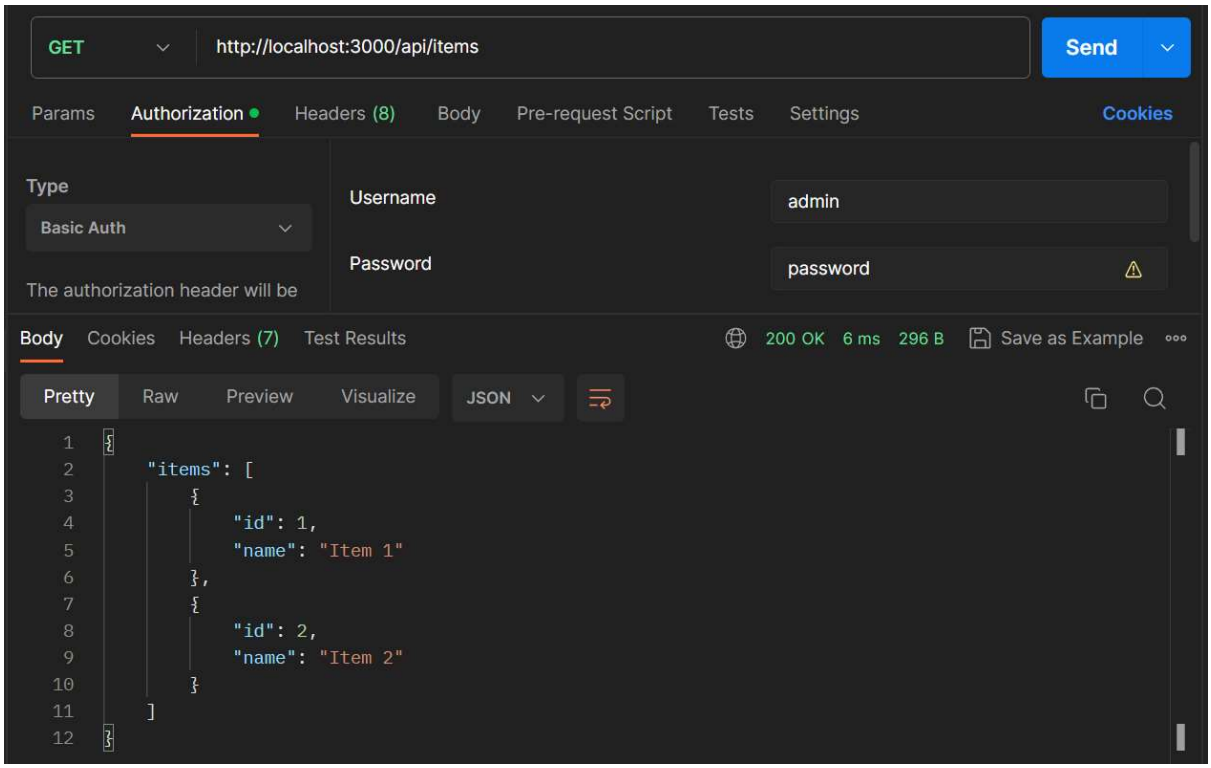
- In the Terminal/CMD , run this following command to run the NodeJS Server :

  `node 5.js`

  ✓ **Output :**

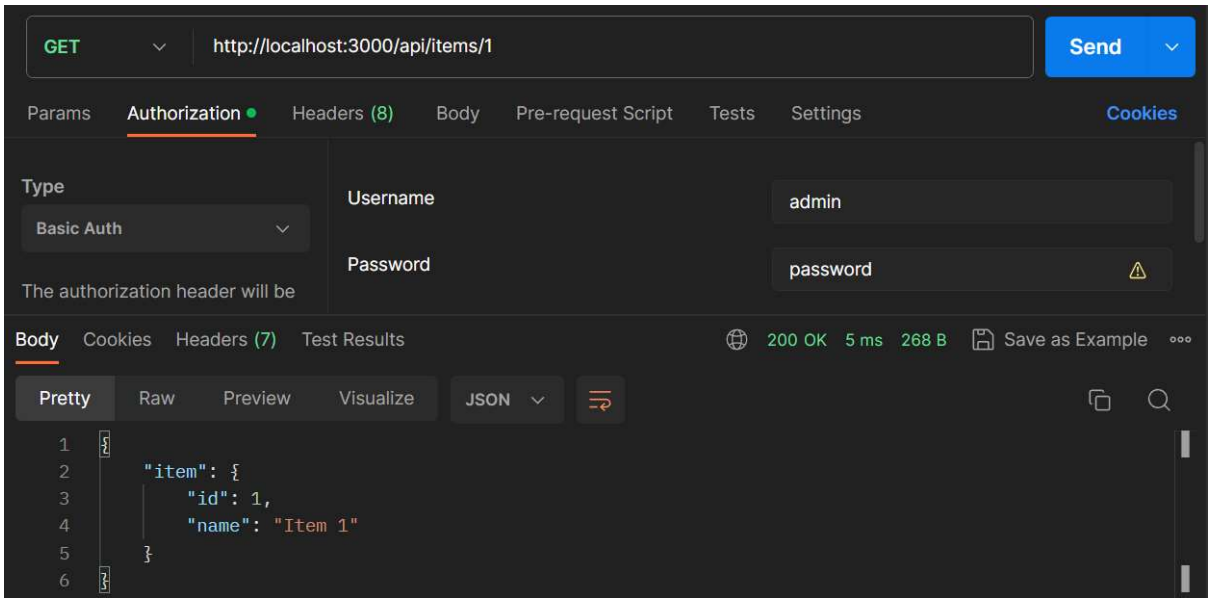
If we run this without authorization it will show Unauthorized and we can't access : **http://localhost:3000/api/items**

→ But if we enter username and password in authorization we can access data :



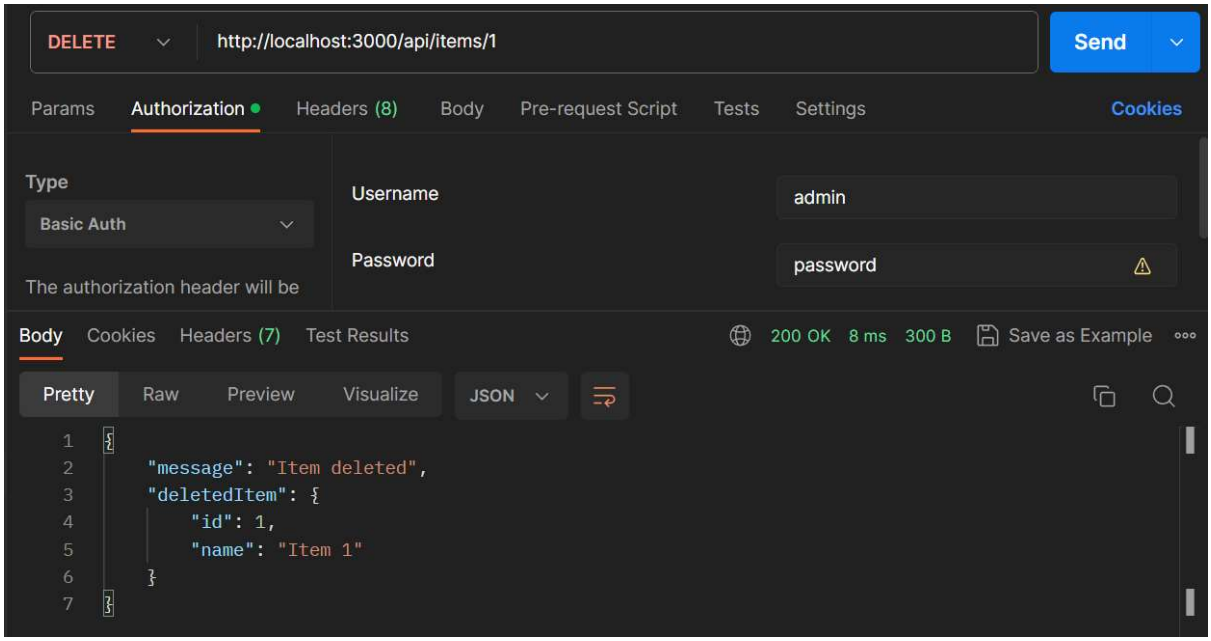
→ FETCHING DATA FROM OUR ITEMS LIST:

http://localhost:3000/api/items/1



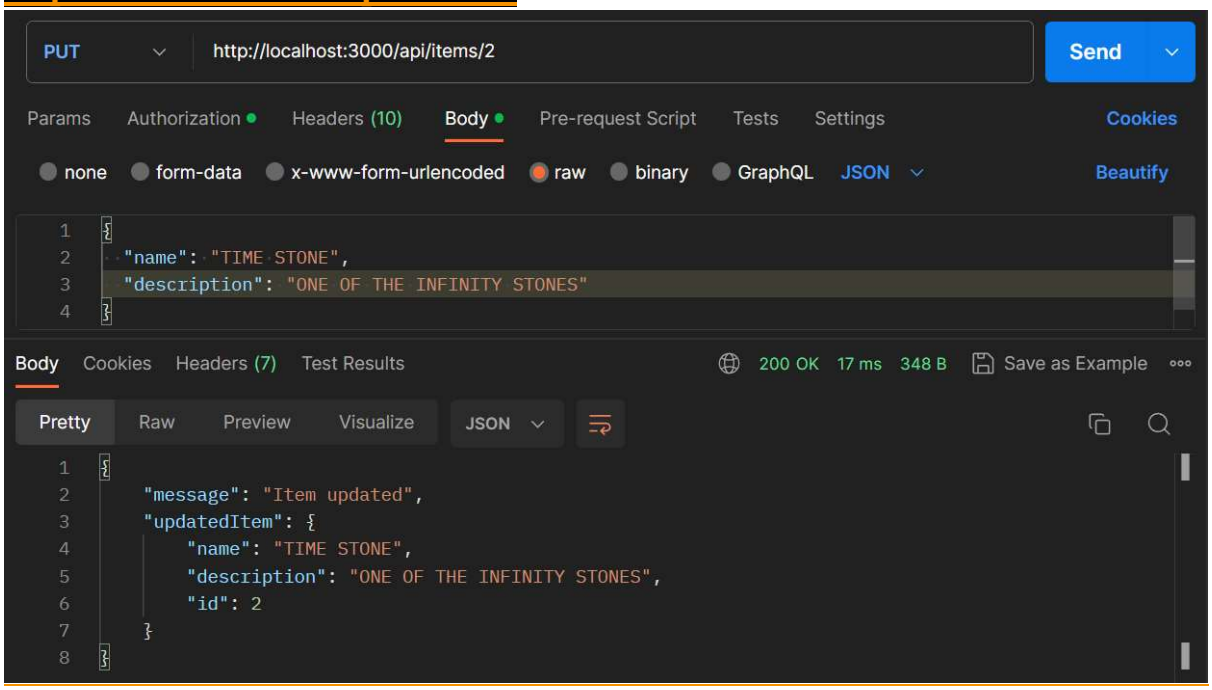
→ DELETING DATA FROM THE ITEMS LIST :

http://localhost:3000/api/items/1



→ UPDATING DATA TO OUR ITEMS LIST :

http://localhost:3000/api/items/2



→ ADDING DATA TO ITEM LIST :

http://localhost:3000/api/items