upGrad

#RahoAmbitious

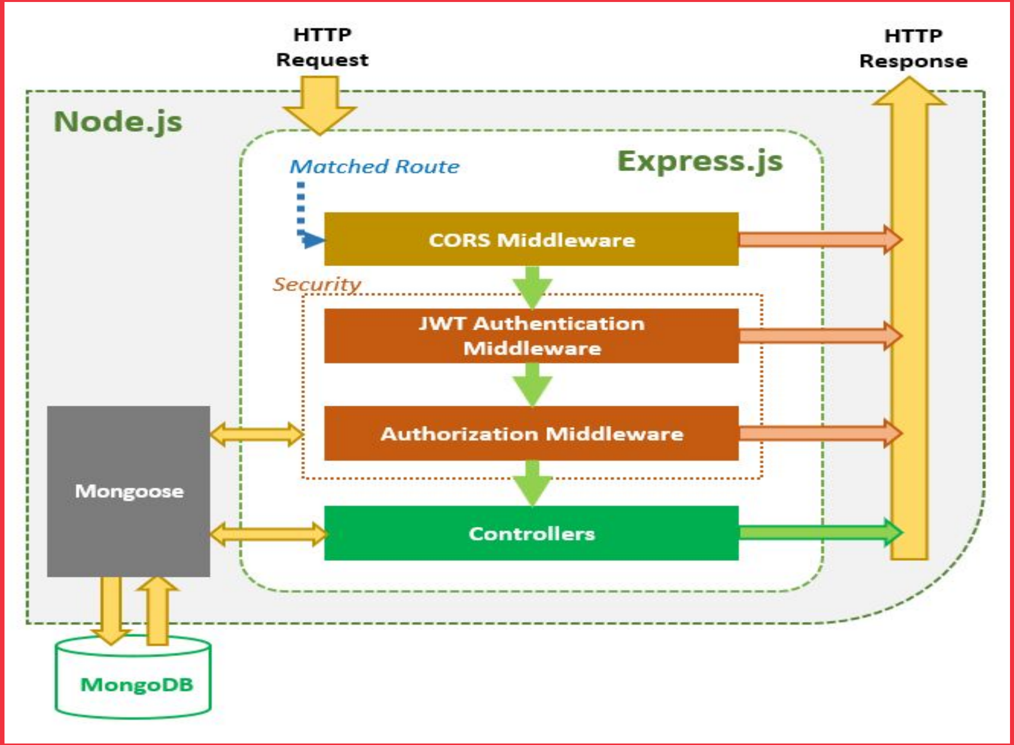
Full Stack Software Development

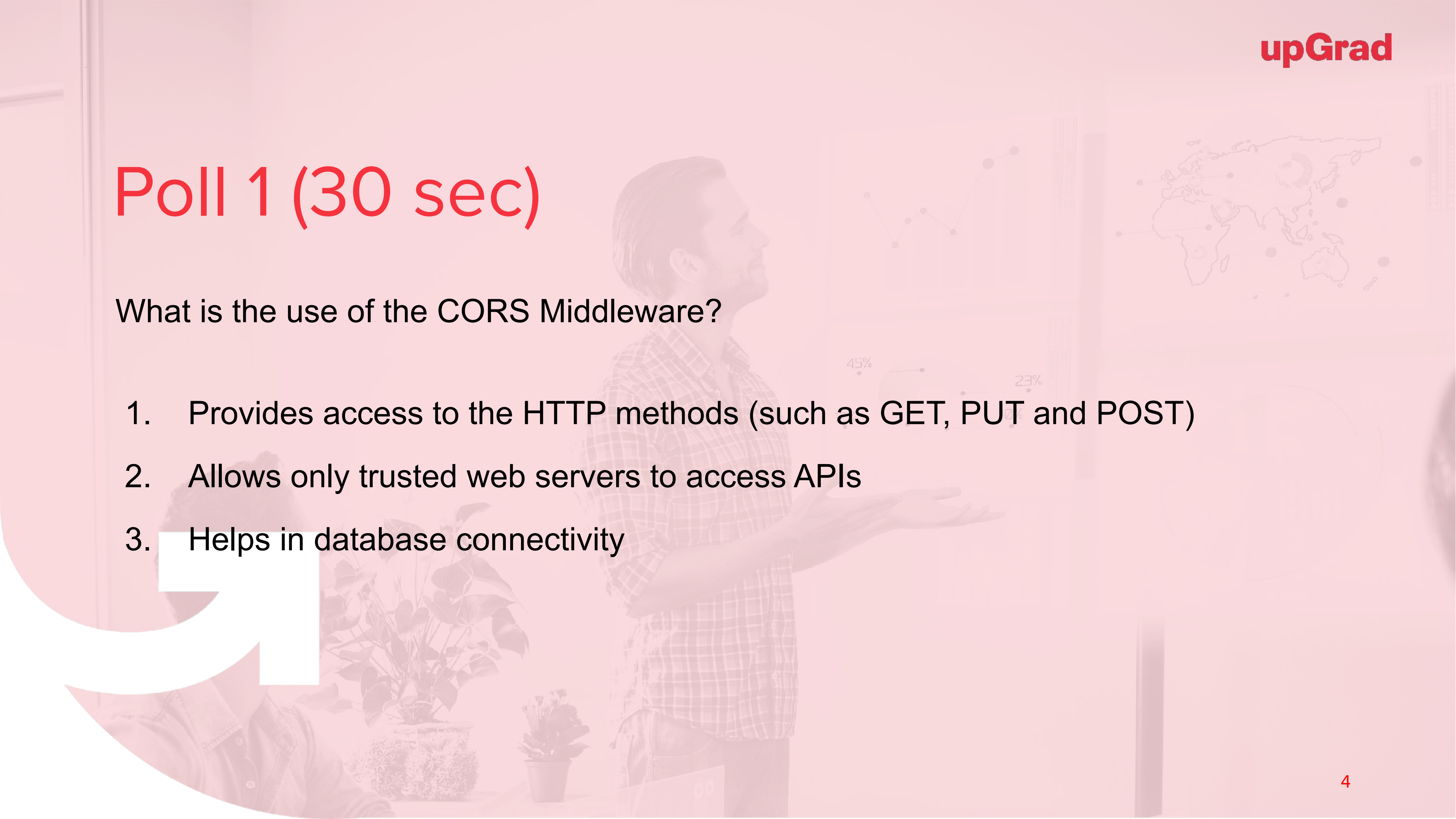
Course: Server-Side Development Using Node.js, Express.js and MongoDB

Lecture on: Authentication/Authorisation

Instructor: Darsh Sanghavi

Authentication/Authorisation Framework



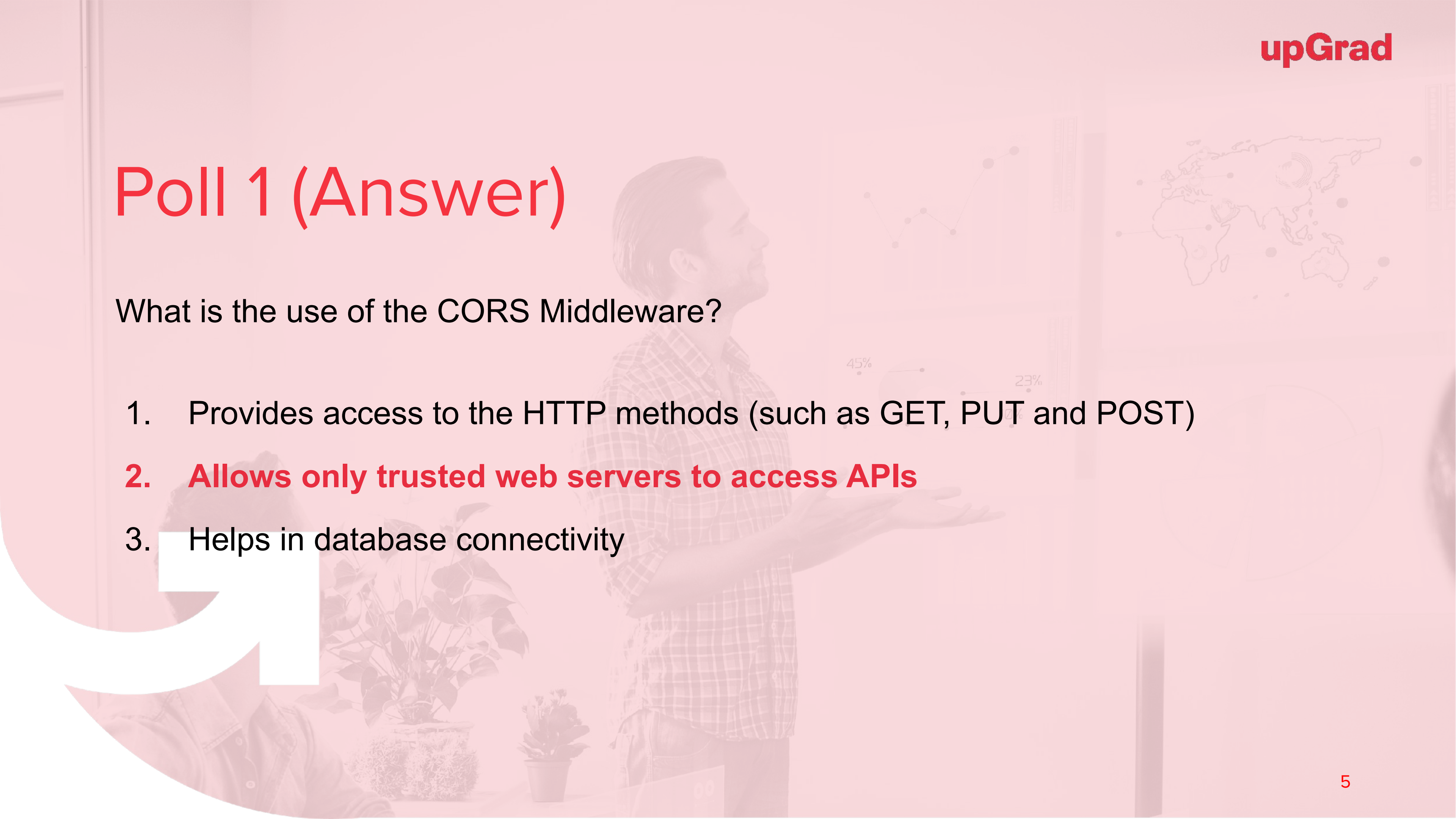


Poll 1 (30 sec)

What is the use of the CORS Middleware?

1. Provides access to the HTTP methods (such as GET, PUT and POST)
2. Allows only trusted web servers to access APIs
3. Helps in database connectivity

Poll 1 (Answer)

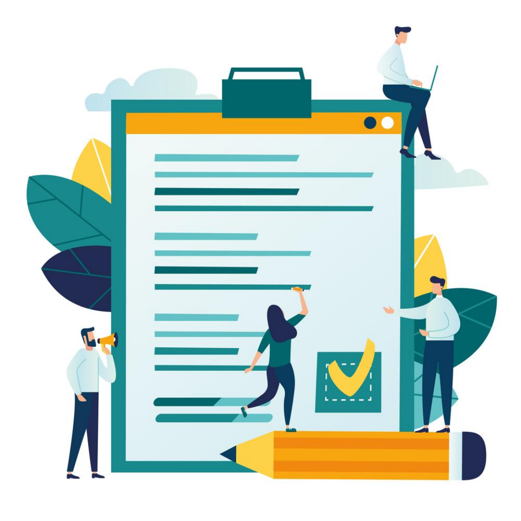


What is the use of the CORS Middleware?

1. Provides access to the HTTP methods (such as GET, PUT and POST)
2. **Allows only trusted web servers to access APIs**
3. Helps in database connectivity

Today's Agenda

- Understand the concept of Authorization
- Understand the concept of Authentication
- Learn how to Authorize using the JWT Package
- Learn how to encrypt data using the Bcrypt Package
- Gain an understanding of Test Authorization using Postman
- Learn about Integrated Authorization and Authentication in our project



Authentication

Need for Authentication/Authorization?

- Consider the example of our edtech project. Suppose we do not have any mechanism to identify users who are authenticated and those who are not

- We will simply provide results to all the users who hit our endpoints (API).

- Without proper authentication/authorization, sensitive information can be compromised.

- So, we need some middleware to bifurcate between system users and any other random users.

- Therefore, it is crucial to authenticate users and provide responses to only system users.

How Can we Authenticate Users?

System users can be identified in different ways. You can create your own functions/middleware where you can generate a secure token, and it can be used to communicate between the server and the client. In this way, we will only serve those clients who have such tokens

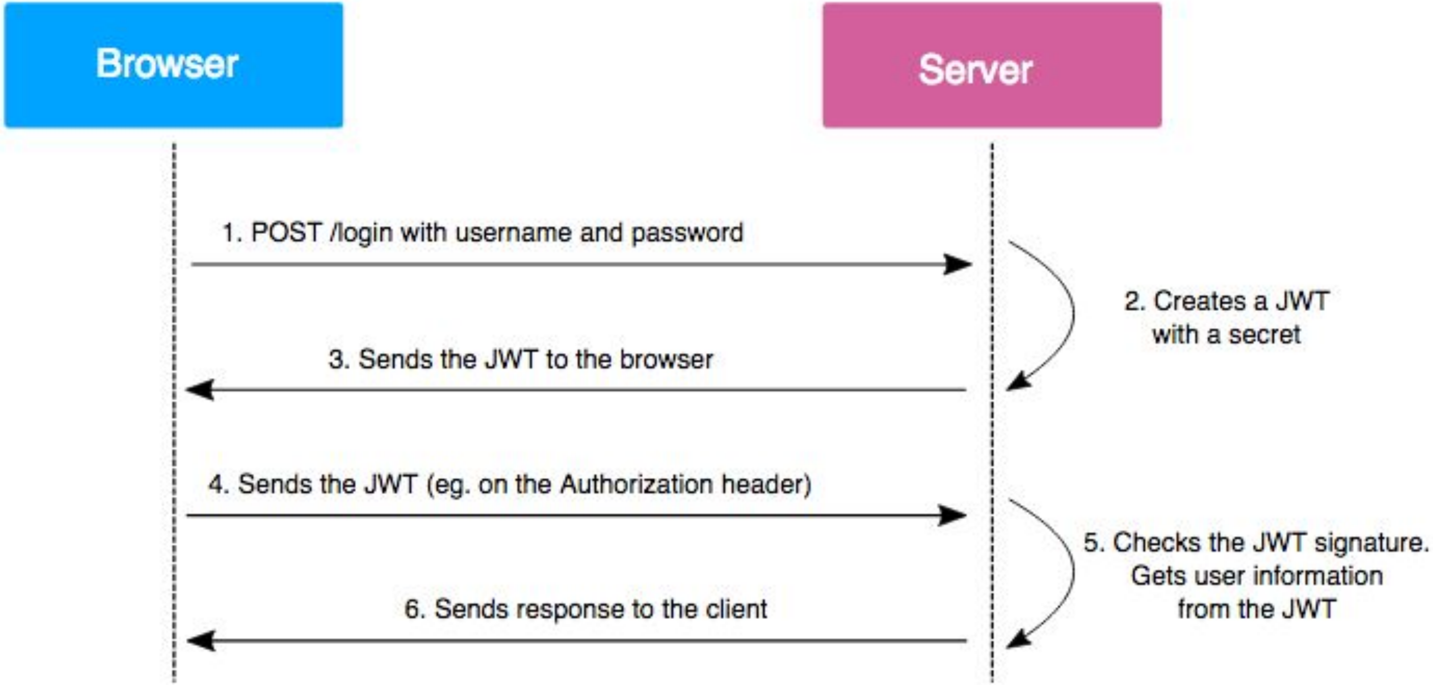
- A more secure one is the JSON Web Tokens (JWT) technique that is also an industry standard and is well accepted by the Node community

- JWT allows you to generate, decode and verify secure tokens by its easy-to-use methods

- Essentially, JWT can be used to authenticate and exchange information between a server and a client

JSON Web Tokens (JWT)

JWT Workflow



Browser
Server
1. POST /login with username and password
2. Creates a JWT with a secret
3. Sends the JWT to the browser
4. Sends the JWT (eg. on the Authorization header)
5. Checks the JWT signature. Gets user information from the JWT
6. Sends response to the client

How Does JWT Work?

1. Client sends credentials to the server
2. Server verifies the credentials, generates a JWT and sends it back as a response
3. Subsequent requests from the client have a JWT in the request headers
4. The server validates the token and, if valid, provides the requested response

Poll 2 (15 sec)

What does JWT stand for?

1. Javascript Web Tokens
2. Java Web Tokens
3. JSON Web Tokens

Poll 2 (Answer)

What does JWT stand for?

1. Javascript Web Tokens
2. Java Web Tokens
3. **JSON Web Tokens**

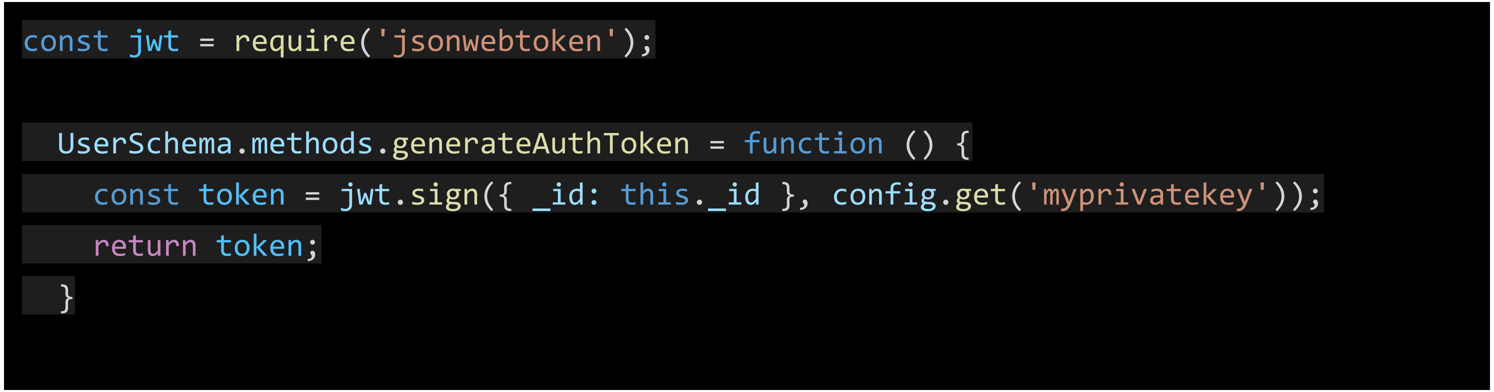
'jsonwebtoken' Package

Implementation of 'jsonwebtoken'

- ***Step 1:***
  Install the jsonwebtoken package from NPM

```
$ npm install jsonwebtoken
```

- ***Step 2:***
  Create a file index.js and set-up express

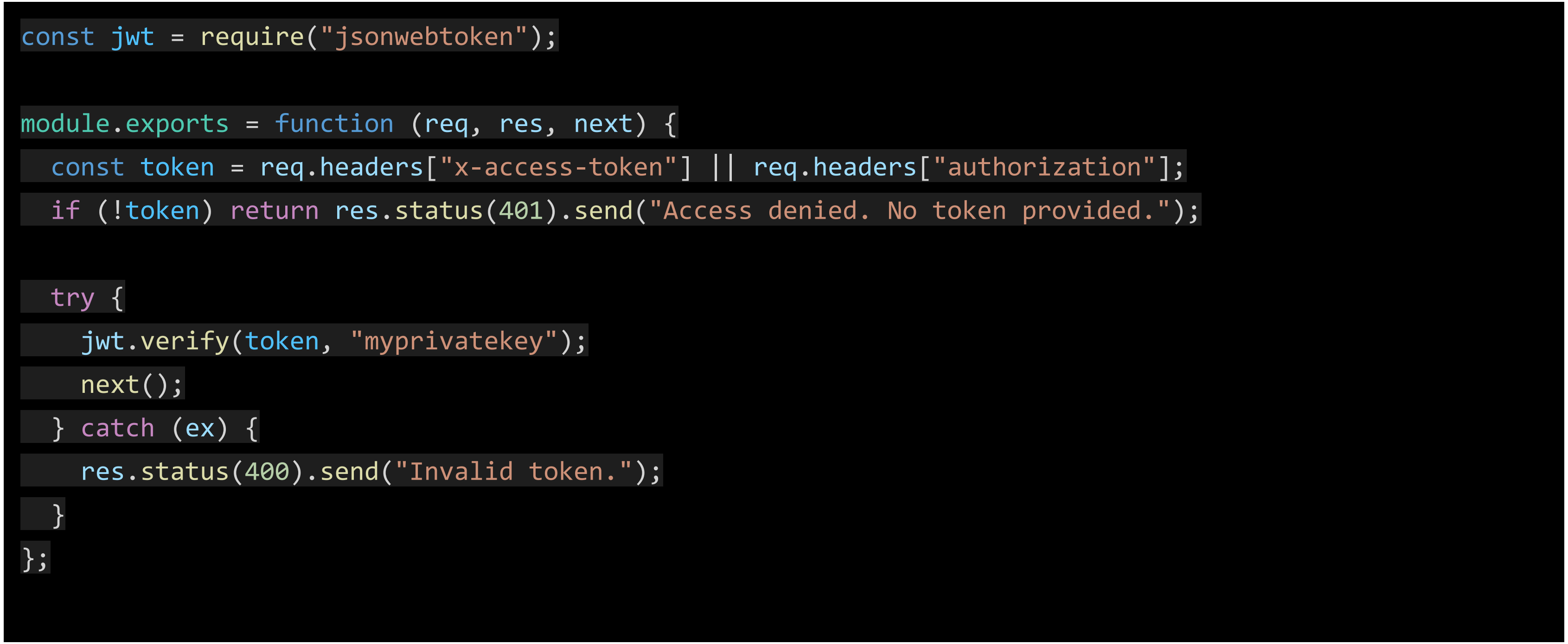


```
const jwt = require('jsonwebtoken');

  UserSchema.methods.generateAuthToken = function () {
    const token = jwt.sign({ _id: this._id }, config.get('myprivatekey'));
    return token;
  }
```

Create a Custom Auth Middleware

Create one file named 'Auth.js' and paste the code given below



```
const jwt = require("jsonwebtoken");

module.exports = function (req, res, next) {
  const token = req.headers["x-access-token"] || req.headers["authorization"];
  if (!token) return res.status(401).send("Access denied. No token provided.");

  try {
    jwt.verify(token, "myprivatekey");
    next();
  } catch (ex) {
    res.status(400).send("Invalid token.");
  }
};
```

How to Use Auth Middleware

- You can use this middleware anywhere in the project and with any/all APIs that we have coded so far.

- Let's use it in our Tutorials Route

```
const auth = require("../middleware/auth");

// Retrieve all Courses by category
    router.get("/category", auth, tutorials.findAllCategory);
```

- As you can see here, we imported our middleware and passed it as a function to our router.

- If the token is valid, the next function will get executed, i.e., `tutorials.findAllCategory`

Doubt Clearance (5 mins)

Key Takeaways

1. Learnt the importance of Authentication/Authorization
2. Learnt how to implement and use the JWT package
3. Coded our custom middleware that uses JWT functions

Tasks to Complete After Today’s Session

Homework
MCQs
Coding Questions

In the Next Class, We Will Discuss...

- Unit Testing and Mocking

upGrad

#RahoAmbitious

Thank You!