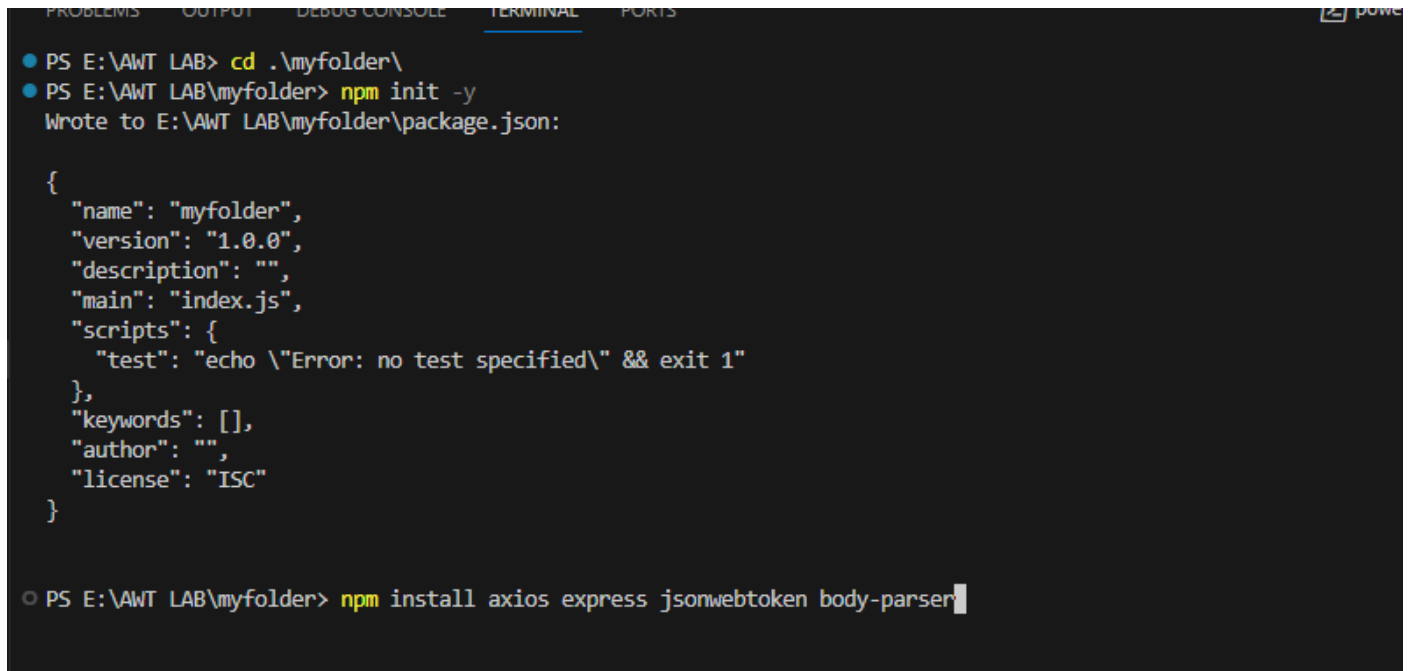


SmallQuestion13

JWT Token Validation: Implement token validation middleware to ensure the integrity of JSON Web Tokens (JWT).

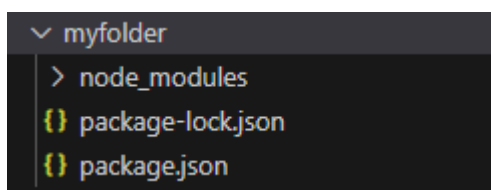
Procedure:

- Create a folder and open that folder in terminal
- Enter the following commands



```
PS E:\AWT LAB> cd .\myfolder\  
PS E:\AWT LAB\myfolder> npm init -y  
Wrote to E:\AWT LAB\myfolder\package.json:  
  
{  
  "name": "myfolder",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "keywords": [],  
  "author": "",  
  "license": "ISC"  
}  
  
PS E:\AWT LAB\myfolder> npm install axios express jsonwebtoken body-parser
```

- The following files and folder will be created in our folder



```
myfolder  
├── node_modules  
├── package-lock.json  
└── package.json
```

- Now create server.js and client.js

server.js



```
const express = require('express');  
const jwt = require('jsonwebtoken');  
const bodyParser = require('body-parser');  
  
const app = express();  
const secretKey = 'yourSecretKey';  
  
// Parse incoming requests with JSON payloads  
app.use(bodyParser.json());
```

```
// Middleware to validate JWT tokens
const validateToken = (req, res, next) => {
  const token = req.header('Authorization');

  // Remove 'Bearer ' from the token if present
  const tokenWithoutBearer = token.replace(/^Bearer /, '');

  if (!tokenWithoutBearer) {
    return res.status(401).json({ error: 'Access denied. Token not provided.' });
  }

  try {
    const decoded = jwt.verify(tokenWithoutBearer, secretKey);
    req.user = decoded;
    next();
  } catch (err) {
    console.log(err)
    return res.status(401).json({ error: 'Invalid token.' });
  }
};

// Protected route example
app.get('/protected-route', validateToken, (req, res) => {
  res.json({ message: 'Access granted. This is a protected route.', user: req.user });
});

// Login route to generate a JWT (replace this with your actual authentication logic)
app.post('/login', (req, res) => {
  const { username, password } = req.body;

  console.log(req.body)

  // Replace this with your actual user authentication logic
  if (username === 'user' && password === 'password') {
    const payload = { username };
    const token = jwt.sign(payload, secretKey, { expiresIn: '1h' });

    res.json({ token });
  } else {
    res.status(401).json({ error: 'Invalid credentials.' });
  }
});

// Start the server
const port = 3000;
app.listen(port, () => {
  console.log(`Server running at http://localhost:${port}/`);
});
```

client.js

```
const axios = require('axios');

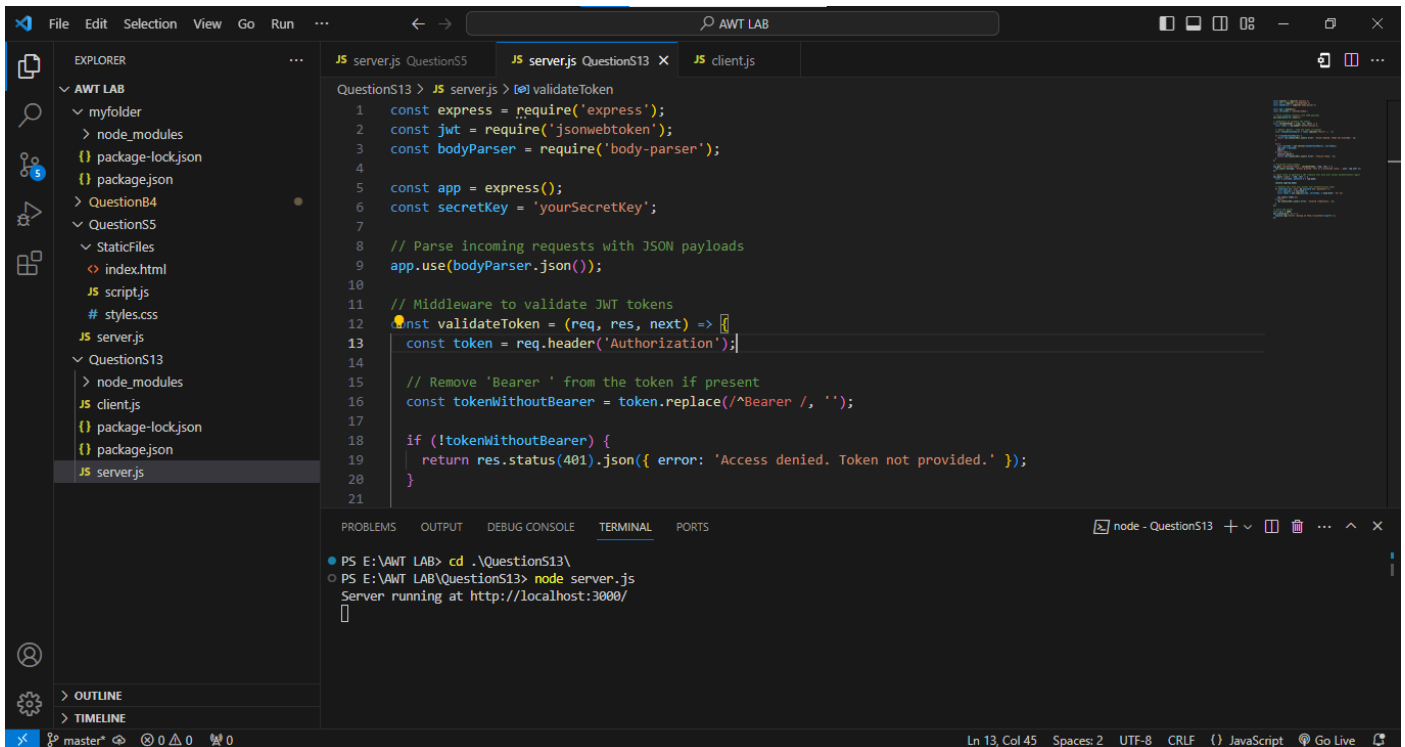
// Replace with your server's URL
const serverUrl = 'http://localhost:3000';

// Replace with your actual credentials for authentication
const credentials = {
  username: 'user',
  password: 'password',
};

// Log in to get the JWT token
axios.post(`${serverUrl}/login`, credentials)
  .then(response => {
    const { token } = response.data;
    console.log(token)

    // Use the token to make a request to the protected route
    axios.get(`${serverUrl}/protected-route`, {
      headers: {
        'Authorization': `Bearer ${token}`
      }
    })
    .then(response => {
      console.log(response.data);
    })
    .catch(error => {
      console.error(error.response.data);
    });
  })
  .catch(error => {
    console.error(error.response.data);
  });
```

- Run server.js

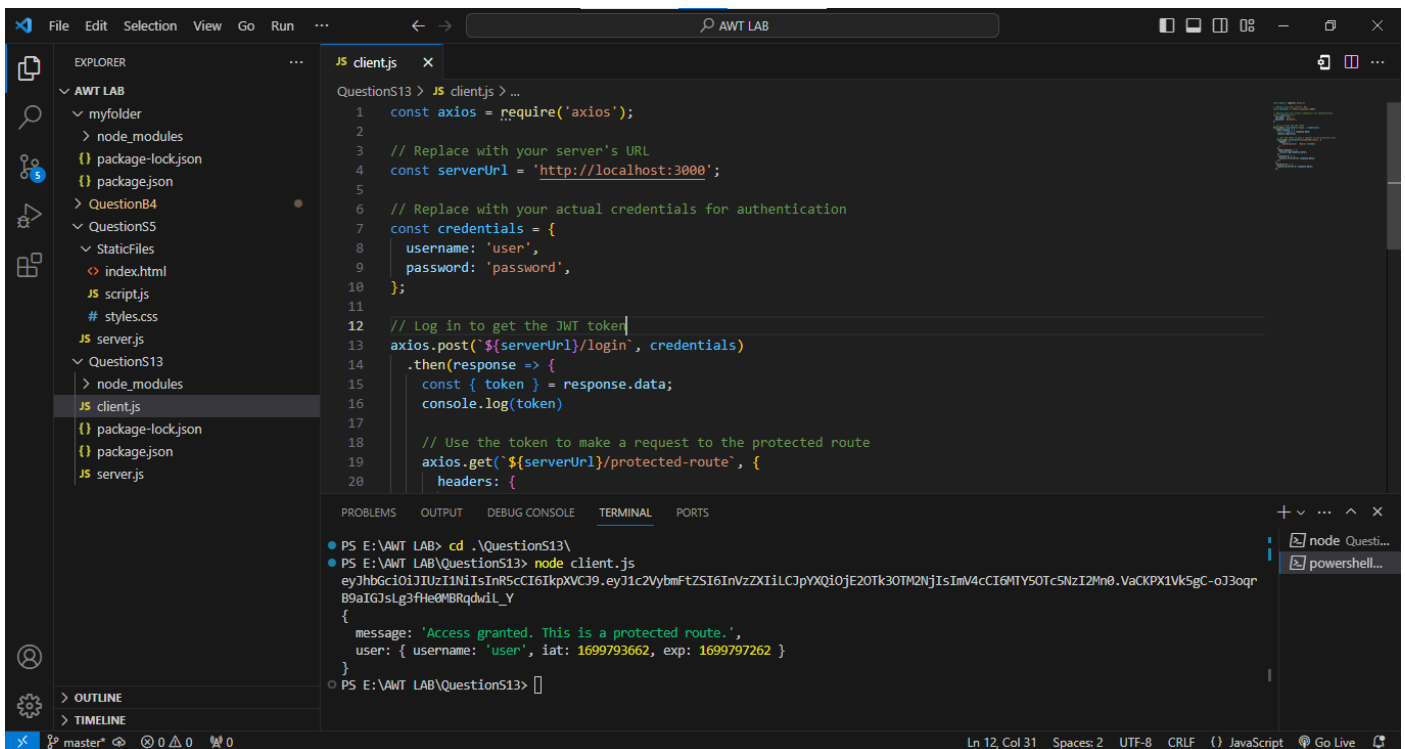


The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'QuestionS13' folder is selected, showing files like 'server.js', 'client.js', 'package-lock.json', and 'package.json'. The main editor displays the content of 'server.js', which is a Node.js application using Express and JWT. The code includes imports for 'express', 'jsonwebtoken', and 'body-parser', followed by the setup of an Express app with a JWT validation middleware. The terminal at the bottom shows the command 'node server.js' being executed, resulting in the message 'Server running at http://localhost:3000/'.

```
QuestionS13 > JS server.js > validateToken
1 const express = require('express');
2 const jwt = require('jsonwebtoken');
3 const bodyParser = require('body-parser');
4
5 const app = express();
6 const secretKey = 'yourSecretKey';
7
8 // Parse incoming requests with JSON payloads
9 app.use(bodyParser.json());
10
11 // Middleware to validate JWT tokens
12 const validateToken = (req, res, next) => {
13   const token = req.header('Authorization');
14
15   // Remove 'Bearer ' from the token if present
16   const tokenWithoutBearer = token.replace(/^Bearer /, '');
17
18   if (!tokenWithoutBearer) {
19     return res.status(401).json({ error: 'Access denied. Token not provided.' });
20   }
21 }
```

PS E:\AWT LAB> cd .\QuestionS13\
PS E:\AWT LAB\QuestionS13> node server.js
Server running at http://localhost:3000/

- Run client.js in another terminal



The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left. The 'QuestionS13' folder is selected, showing files like 'server.js', 'client.js', 'package-lock.json', and 'package.json'. The main editor displays the content of 'client.js', which is a Node.js application using Axios to make HTTP requests. The code includes imports for 'axios', followed by the setup of a server URL and credentials. It then uses 'axios.post' to log in and 'axios.get' to fetch data from a protected route. The terminal at the bottom shows the command 'node client.js' being executed, resulting in a JSON response with a message and user information.

```
QuestionS13 > JS client.js > ...
1 const axios = require('axios');
2
3 // Replace with your server's URL
4 const serverUrl = 'http://localhost:3000';
5
6 // Replace with your actual credentials for authentication
7 const credentials = {
8   username: 'user',
9   password: 'password',
10 };
11
12 // Log in to get the JWT token
13 axios.post(`${serverUrl}/login`, credentials)
14   .then(response => {
15     const { token } = response.data;
16     console.log(token)
17
18     // Use the token to make a request to the protected route
19     axios.get(`${serverUrl}/protected-route`, {
20       headers: {

```

PS E:\AWT LAB> cd .\QuestionS13\
PS E:\AWT LAB\QuestionS13> node client.js
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImV4dCJpYXQiOiJlY20tMj02MjIjIiwiaWV4cCI6IjY5OTc5NzI2Mn0.VaCKPX1V5gC-o33oqrB9aIG3sLg3fHeQMBRqdwil_Y
{
 message: 'Access granted. This is a protected route.',
 user: { username: 'user', iat: 1699793662, exp: 1699797262 }
}
PS E:\AWT LAB\QuestionS13>