

Lesson 09 Demo 02

Using Middleware in Express.js App

Objective: To use middleware in the Express.js app for enhanced request handling and functionality

Tools Required: Visual Studio, Node.js, and Express.js

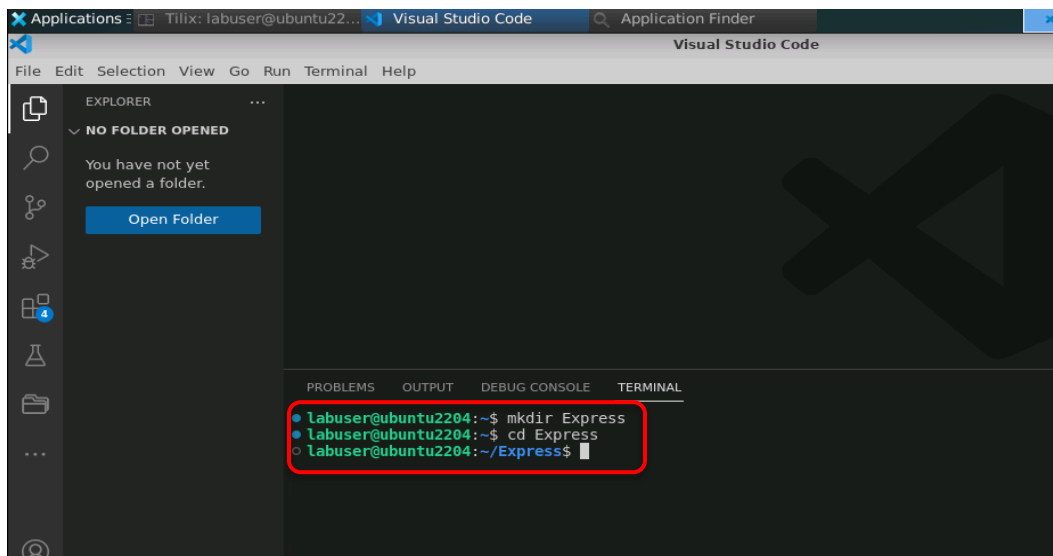
Prerequisites: Knowledge of JavaScript and Node.js

Steps to be followed:

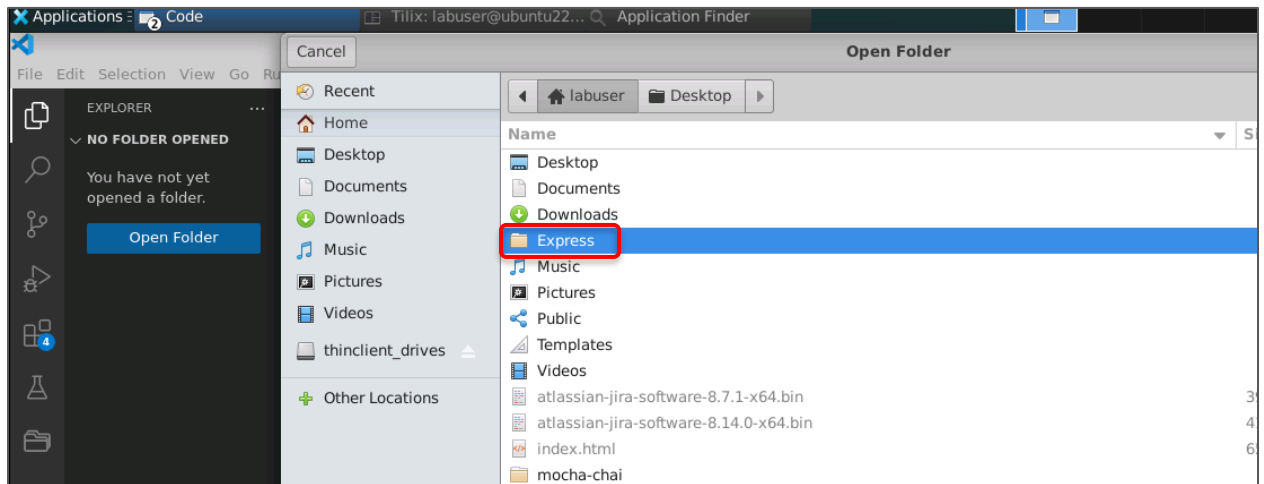
1. Set up Express.js
2. Demonstrate application-level middleware in Express.js
3. Demonstrate router-level middleware in Express.js
4. Demonstrate error-handling middleware in Express.js
5. Demonstrate third-party middleware in Express.js

Step 1: Set up Express.js

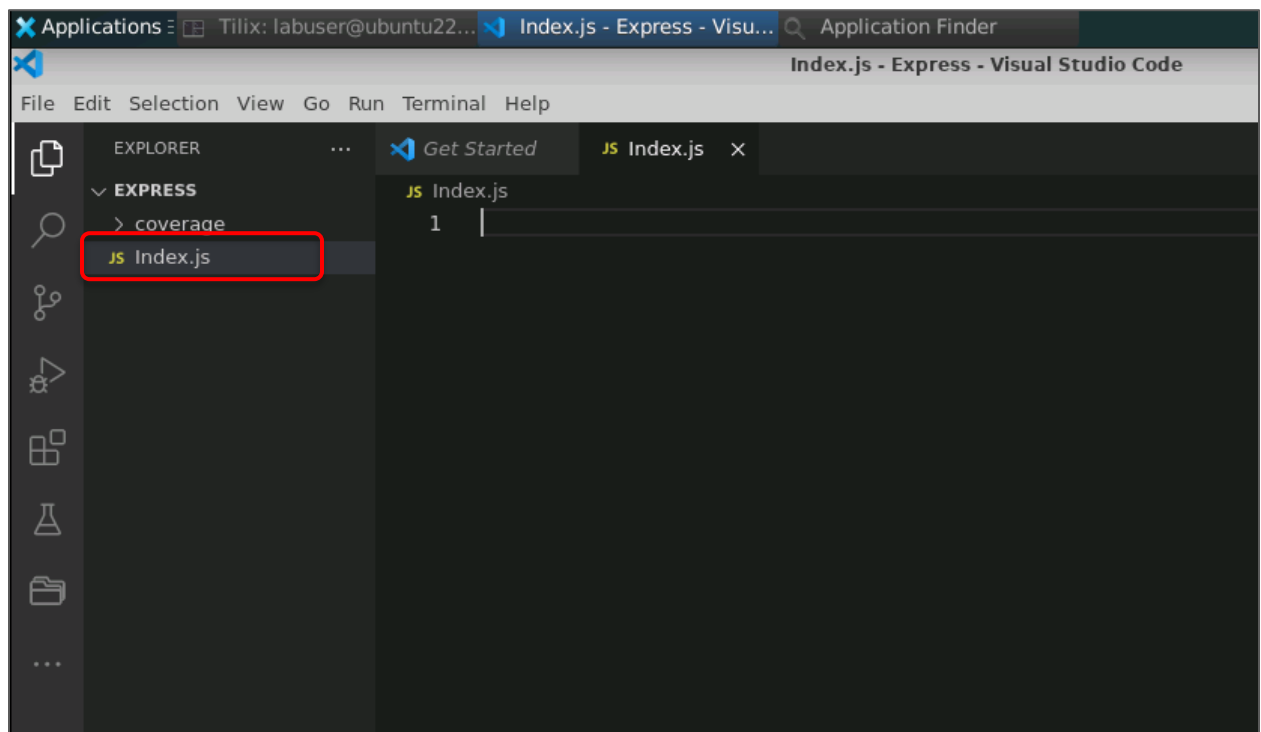
- 1.1 Open VS Code, create a folder named Express.js with the command **mkdir Express**, and change the current working directory using **cd Express**



1.2 Open the **Express** folder in the VS Code



1.3 Create an **index.html** file

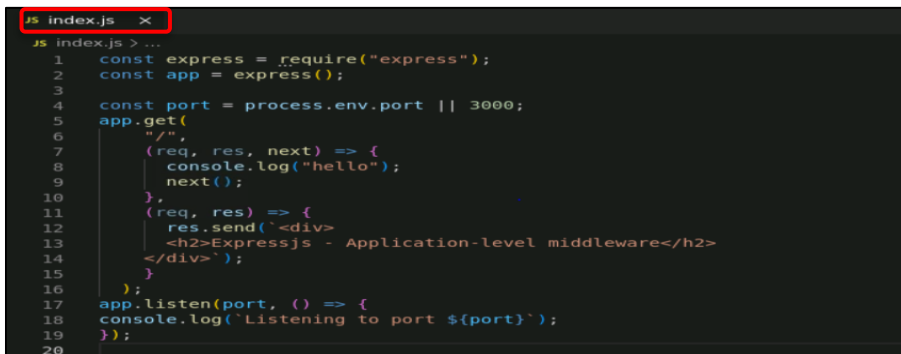


Step 2: Demonstrate application-level middleware in Express.js

2.1 Add the following code in the `index.js` file:

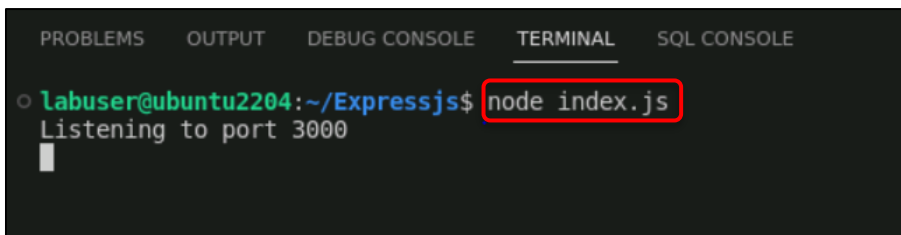
```
const express = require("express");
const app = express();

const port = process.env.port || 3000;
app.get(
  "/",
  (req, res, next) => {
    console.log("hello");
    next();
  },
  (req, res) => {
    res.send(`<div>
    <h2>Expressjs - Application-level middleware</h2>
    </div>`);
  }
);
app.listen(port, () => {
  console.log(`Listening to port ${port}`);
});
```

A screenshot of a code editor window titled 'js index.js'. The code inside is the same as the one in the previous block, showing the setup of an Express.js application with a single GET route for the root path. The code is as follows:

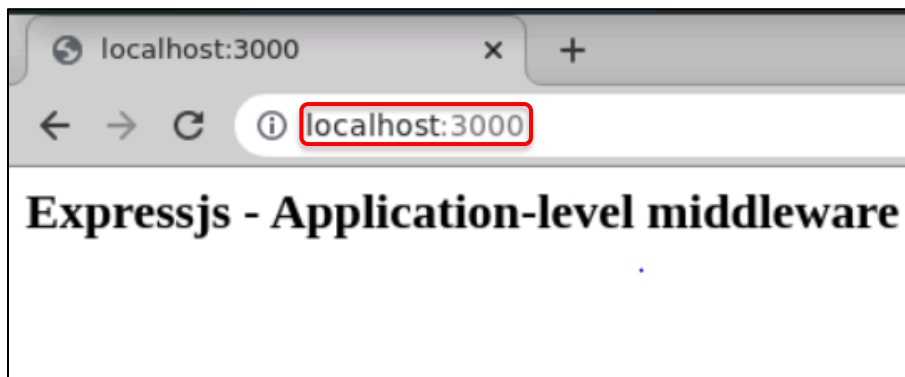
```
1 const express = require("express");
2 const app = express();
3
4 const port = process.env.port || 3000;
5 app.get(
6   "/",
7   (req, res, next) => {
8     console.log("hello");
9     next();
10  },
11  (req, res) => {
12    res.send(`<div>
13    <h2>Expressjs - Application-level middleware</h2>
14    </div>`);
15  }
16 );
17 app.listen(port, () => {
18   console.log(`Listening to port ${port}`);
19 });
20
```

2.2 Run the `node index.js` command in the terminal and get the output

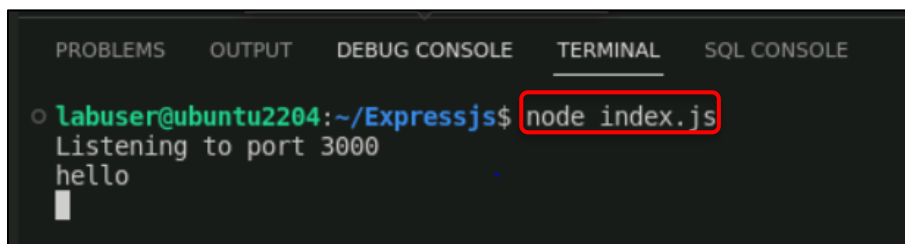
A screenshot of a terminal window with tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL', and 'SQL CONSOLE'. The 'TERMINAL' tab is active. The prompt shows the user is 'labuser@ubuntu2204' in the directory '~/Expressjs'. The command 'node index.js' has been entered and is highlighted with a red box. The output of the command is 'Listening to port 3000', followed by a cursor on a new line.

```
labuser@ubuntu2204:~/Expressjs$ node index.js
Listening to port 3000
```

2.3 Open the browser and browse <http://localhost:3000>



2.4 Check the terminal



Step 3: Demonstrate router-level middleware in Express.js

3.1 Write the following code in the `index.js` file:

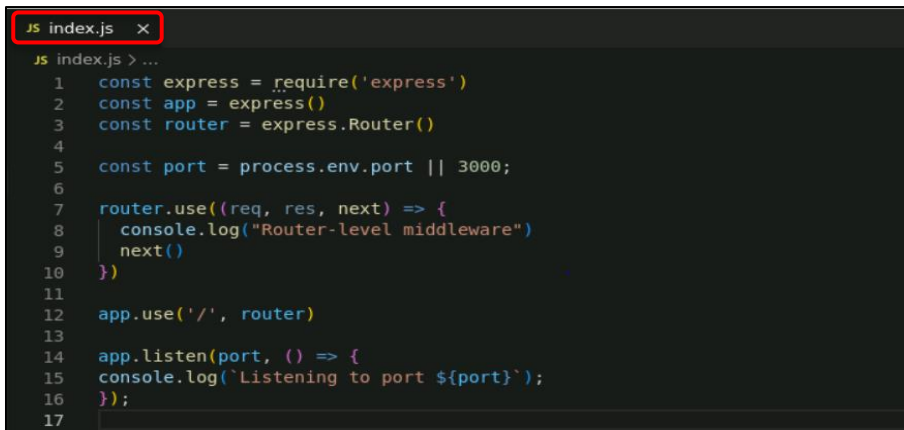
```
const express = require('express')
const app = express()
const router = express.Router()

const port = process.env.port || 3000;

router.use((req, res, next) => {
  console.log("Router-level middleware")
  next()
})

app.use('/', router)

app.listen(port, () => {
  console.log(`Listening to port ${port}`);
});
```

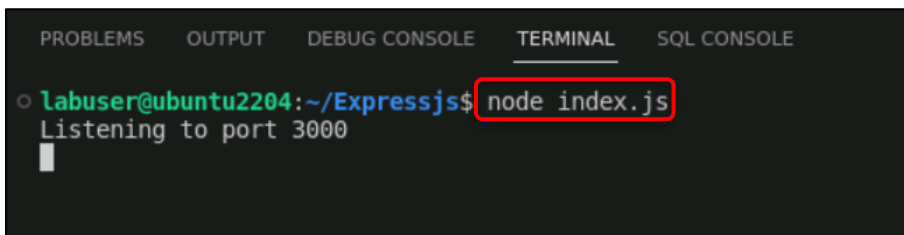


```

JS index.js x
JS index.js > ...
1  const express = require('express')
2  const app = express()
3  const router = express.Router()
4
5  const port = process.env.port || 3000;
6
7  router.use((req, res, next) => {
8    console.log("Router-level middleware")
9    next()
10 })
11
12 app.use('/', router)
13
14 app.listen(port, () => {
15   console.log(`Listening to port ${port}`);
16 });
17

```

3.2 Run the **node index.js** command in the terminal and verify the output

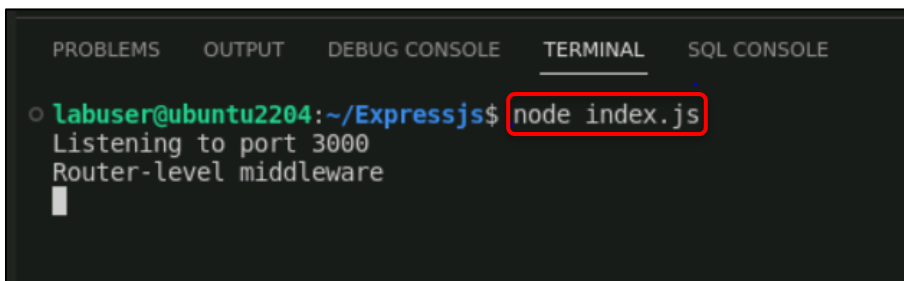


```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Listening to port 3000

```

3.3 Open the browser, browse <http://localhost:3000>, and check the terminal again



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Listening to port 3000
Router-level middleware

```

Step 4: Demonstrate error-handling middleware in Express.js

4.1 Add the following code in the **index.js** file:

```

const express = require('express')
const app = express()

const port = process.env.port || 3000;

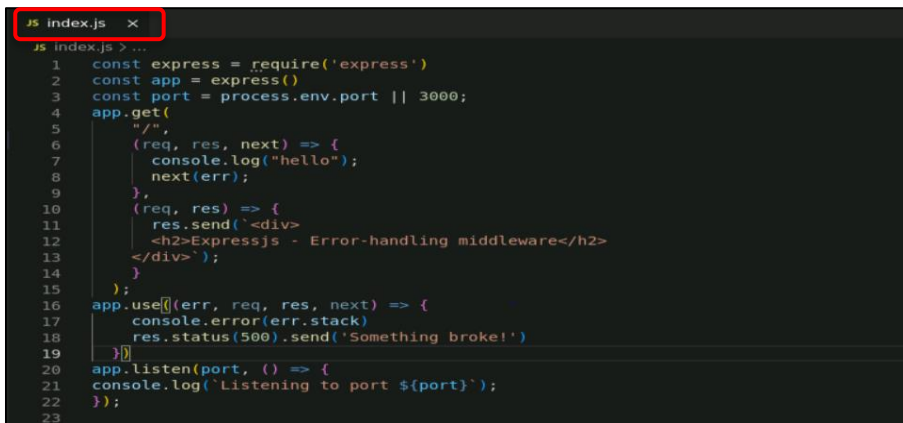
app.get(
  "/",
  (req, res, next) => {
    console.log("hello");

```

```
    next(err);
  },
  (req, res) => {
    res.send(`<div>
      <h2>Expressjs - Error-handling middleware</h2>
    </div>`);
  }
);

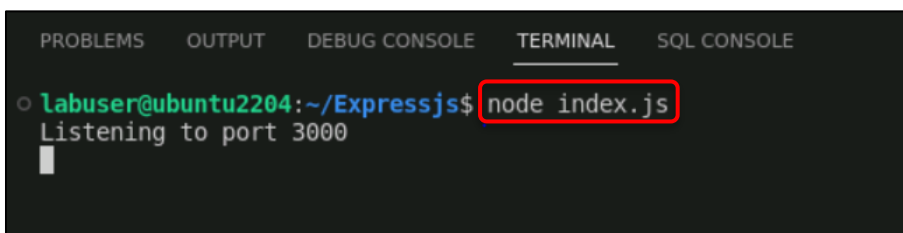
app.use((err, req, res, next) => {
  console.error(err.stack)
  res.status(500).send('Something broke!')
})

app.listen(port, () => {
  console.log(`Listening to port ${port}`);
});
```



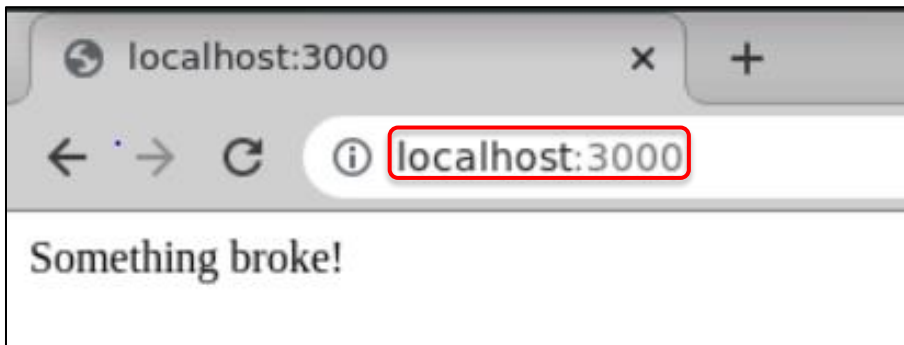
```
1  const express = require('express')
2  const app = express()
3  const port = process.env.port || 3000;
4  app.get(
5    "/",
6    (req, res, next) => {
7      console.log("hello");
8      next(err);
9    },
10   (req, res) => {
11     res.send(`<div>
12       <h2>Expressjs - Error-handling middleware</h2>
13     </div>`);
14   }
15 );
16 app.use((err, req, res, next) => {
17   console.error(err.stack)
18   res.status(500).send('Something broke!')
19 })
20 app.listen(port, () => {
21   console.log(`Listening to port ${port}`);
22 });
23
```

4.2 Run the **node index.js** command in the terminal and get the output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Listening to port 3000
```

4.3 Open the browser, browse <http://localhost:3000>, and check the terminal again



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Listening to port 3000
hello
ReferenceError: err is not defined
    at /home/labuser/Expressjs/index.js:10:12
    at Layer.handle [as handle_request] (/home/labuser/Expressjs/node_modules/express/lib/router/layer.js:95:5)
    at next (/home/labuser/Expressjs/node_modules/express/lib/router/route.js:144:13)
    at Route.dispatch (/home/labuser/Expressjs/node_modules/express/lib/router/route.js:114:3)
    at Layer.handle [as handle_request] (/home/labuser/Expressjs/node_modules/express/lib/router/layer.js:95:5)
    at /home/labuser/Expressjs/node_modules/express/lib/router/index.js:284:15
    at Function.process_params (/home/labuser/Expressjs/node_modules/express/lib/router/index.js:346:12)
    at next (/home/labuser/Expressjs/node_modules/express/lib/router/index.js:280:10)
    at expressInit (/home/labuser/Expressjs/node_modules/express/lib/middleware/init.js:40:5)
    at Layer.handle [as handle_request] (/home/labuser/Expressjs/node_modules/express/lib/router/layer.js:95:5)

```

Step 5: Demonstrate third-party middleware in Express.js

5.1 Install third-party middleware, use the following command, and run it in the terminal:
npm install cookie-parser

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ npm install cookie-parser
added 2 packages, and audited 60 packages in 1s

7 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
labuser@ubuntu2204:~/Expressjs$

```

5.2 Add the following code in the **index.js** file, which uses third-party middleware:

```


const express = require('express')
const app = express()
const cookieParser = require('cookie-parser')

const port = process.env.port || 3000;

```

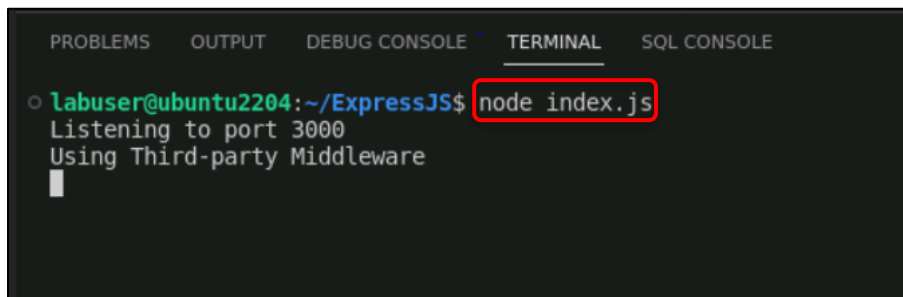
```
// load the cookie-parsing middleware
app.use(cookieParser())

app.listen(port, () => {
  console.log(`Listening to port ${port}`);
  console.log('Using Third-party Middleware.')
});
```



```
JS index.js x
JS index.js > ...
1  const express = require('express')
2  const app = express()
3  const cookieParser = require('cookie-parser')
4
5  const port = process.env.port || 3000;
6  // load the cookie-parsing middleware
7  app.use(cookieParser())
8
9  app.listen(port, () => {
10     console.log(`Listening to port ${port}`);
11     console.log('Using Third-party Middleware')
12   });
13
```

5.3 Run the **node index.js** command in the terminal and verify the output



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/ExpressJS$ node index.js
Listening to port 3000
Using Third-party Middleware
```

By following these steps, you have effectively utilized diverse middleware in an **Express.js** app to enhance request handling and functionality.