

## Lesson 08 Demo 03

### Working with Express.js Frameworks

**Objective:** To work with Express.js framework for a comprehensive understanding of web development

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

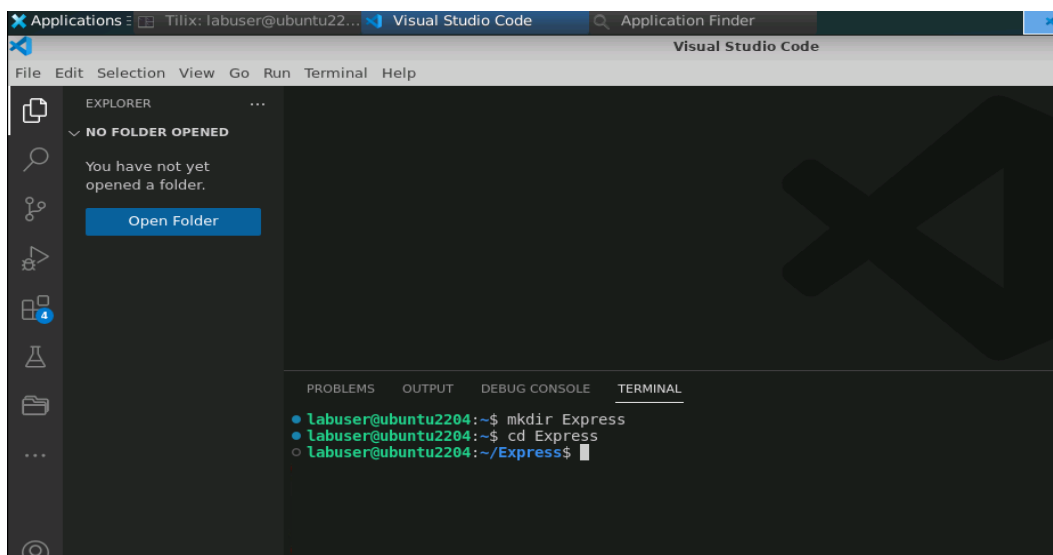
**Prerequisites:** Knowledge of JavaScript and Node.js

Steps to be followed:

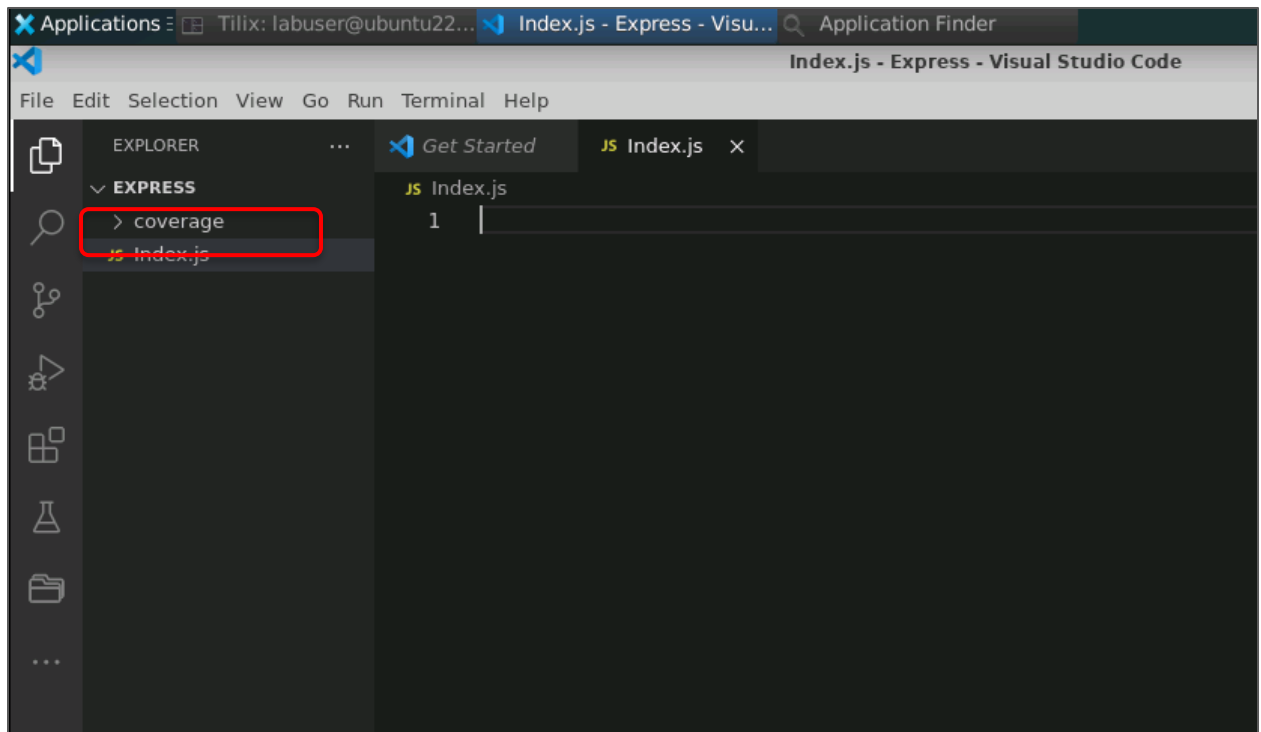
1. Perform routing in Express.js
2. Build URL in Express.js
3. Template in Express.js
4. Create StaticFiles in Express.js
5. Handle Error States in Express.js
6. Debug in Express.js Module

#### Step 1: Perform routing in Express.js

- 1.1 Open the VS code and create a folder name **ExpressJS** using the command **mkdir** and then use **cd Express** to change the current working directory



1.2 Open the **Express** folder in VS code and create an **index.html** file



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

```
var express = require('express');
var app = express();
var PORT = 3000;
// Single routing
var router = express.Router();
router.get('/', function (req, res, next) {
  console.log("express.Router() is Working");
  res.end();
})
app.use(router);
app.listen(PORT, function(err){
  if (err) console.log(err);
  console.log("Server listening on PORT", PORT);
});
```

```

1  var express = require('express');
2  var app = express();
3  var PORT = 3000;
4  // Single routing
5  var router = express.Router();
6  router.get('/', function (req, res, next) {
7    console.log("express.Router() is Working");
8    res.end();
9  });
10 app.use(router);
11 app.listen(PORT, function(err){
12   if (err) console.log(err);
13   console.log("Server listening on PORT", PORT);
14 });
15

```

1.4 Run the **node index.js** command in the terminal

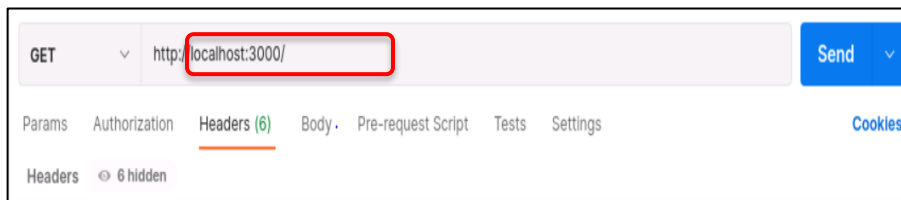
```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000

```

1.5 Make a GET request to **http://localhost:3000/**, and check the output in vs code terminal, where the program is executed



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000
express.Router() is Working

```

## Step 2: Build URL in Express.js

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

```

const express = require('express');
const app = express();
const PORT = 3000;

```

```

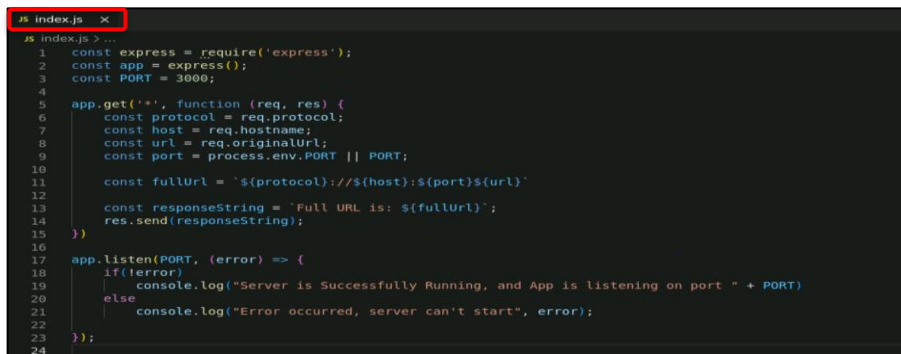
app.get('*', function (req, res) {
  const protocol = req.protocol;
  const host = req.hostname;
  const url = req.originalUrl;
  const port = process.env.PORT || PORT;

  const fullUrl = `${protocol}://${host}:${port}${url}`

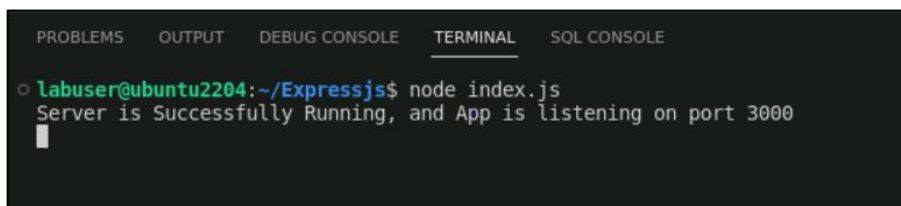
  const responseString = `Full URL is: ${fullUrl}`;
  res.send(responseString);
})

app.listen(PORT, (error) => {
  if(!error)
    console.log("Server is Successfully Running, and App is listening on port " +
PORT)
  else
    console.log("Error occurred, server can't start", error);
});

```



## 2.2 Run the **node index.js** command in the terminal

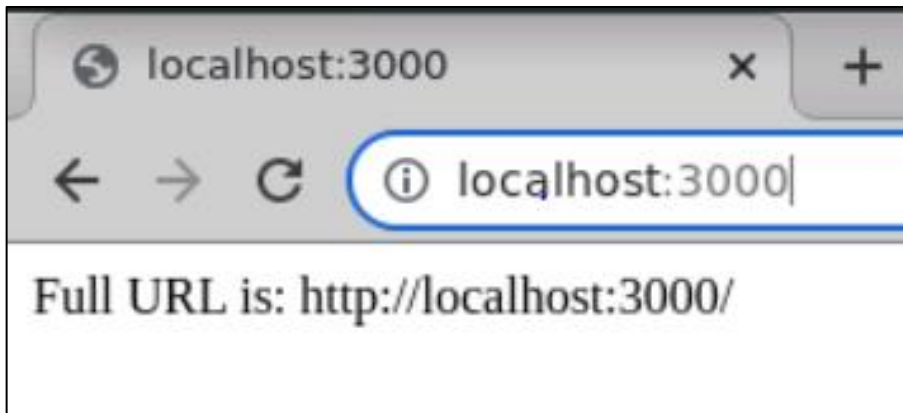


```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Server is Successfully Running, and App is listening on port 3000

```

2.3 Browse for <http://localhost:3000/>



### Step 3: Template in Express.js:

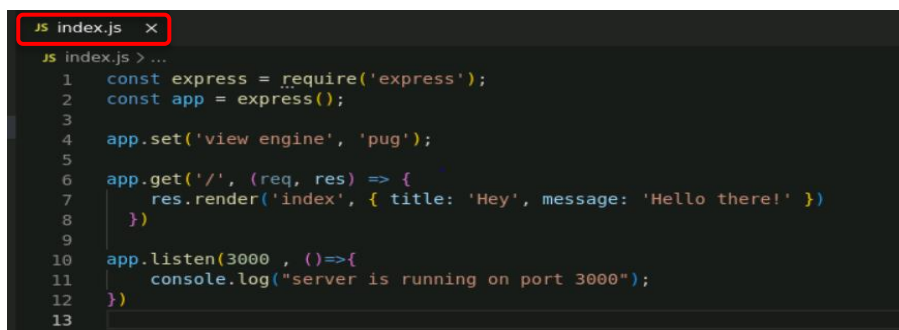
3.1 Add the following code in the **index.js** file:

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

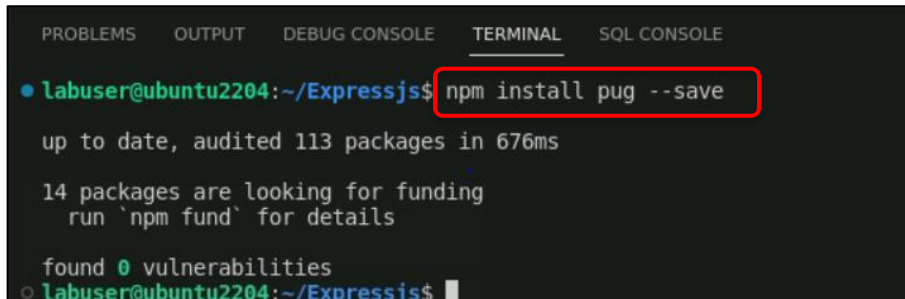
app.set('view engine', 'pug');

app.get('/', (req, res) => {
  res.render('index', { title: 'Hey', message: 'Hello there!' })
})

app.listen(3000, ()=>{
  console.log("server is running on port 3000");
})
```



3.2 Run the following command in the terminal to install the **pug engine**  
**npm install pug --save**



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE

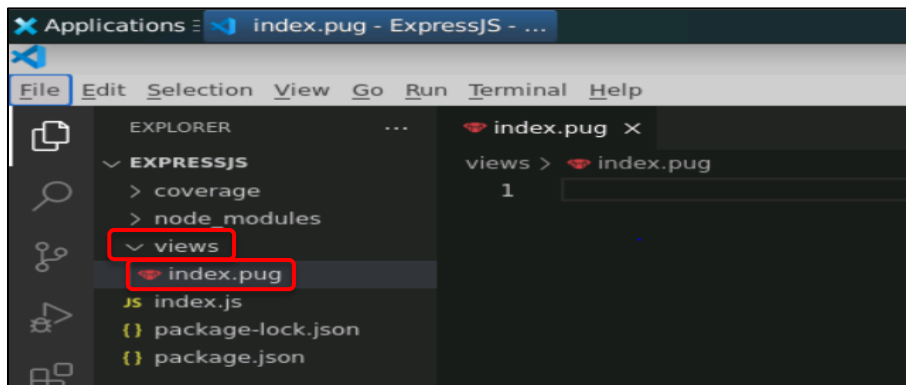
• labuser@ubuntu2204:~/Expressjs$ npm install pug --save

up to date, audited 113 packages in 676ms

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

found 0 vulnerabilities
labuser@ubuntu2204:~/Expressjs$
  
```

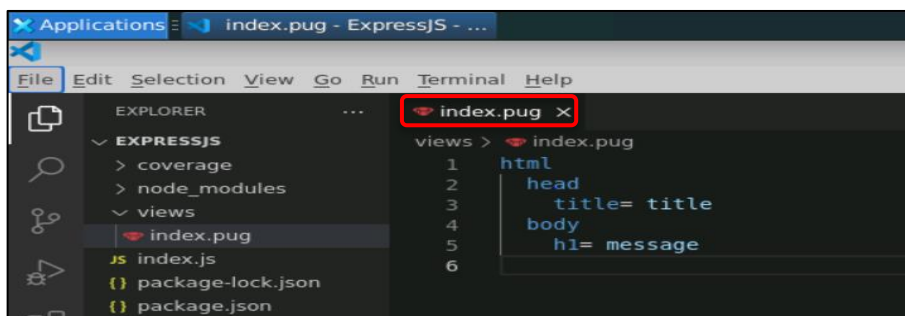
3.3 Create the **views** name folder, and inside it, create the **index.pug** file



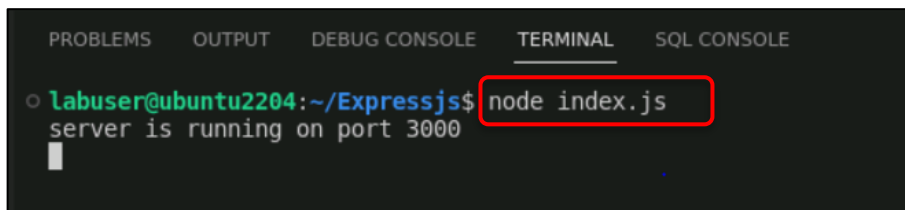
3.4 Add following code in **index.pug** file:

```

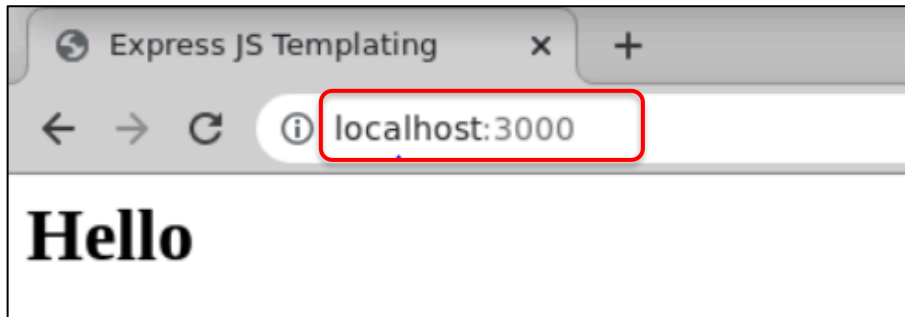
html
head
  title= title
body
  h1= message
  
```



3.5 Run the **node index.js** command in the terminal and browse for **http://localhost:3000/**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
server is running on port 3000
```

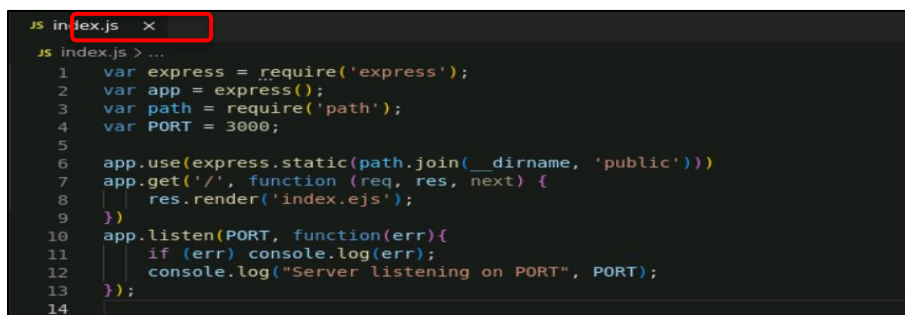


#### Step 4: Create StaticFiles in Express.js:

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

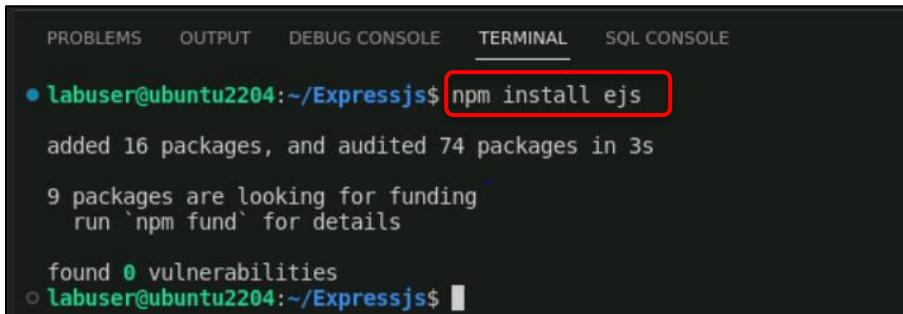
```
var express = require('express');
var app = express();
var path = require('path');
var PORT = 3000;

app.use(express.static(path.join(__dirname, 'public')))
app.get('/', function (req, res, next) {
  res.render('index.ejs');
})
app.listen(PORT, function(err){
  if (err) console.log(err);
  console.log("Server listening on PORT", PORT);
});
```



```
JS index.js x
JS index.js > ...
1 var express = require('express');
2 var app = express();
3 var path = require('path');
4 var PORT = 3000;
5
6 app.use(express.static(path.join(__dirname, 'public')))
7 app.get('/', function (req, res, next) {
8   res.render('index.ejs');
9 })
10 app.listen(PORT, function(err){
11   if (err) console.log(err);
12   console.log("Server listening on PORT", PORT);
13 });
14
```

- 4.2 Run the command below in the terminal to add ejs to the project:  
**npm install ejs**



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ npm install ejs
added 16 packages, and audited 74 packages in 3s
9 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
labuser@ubuntu2204:~/Expressjs$

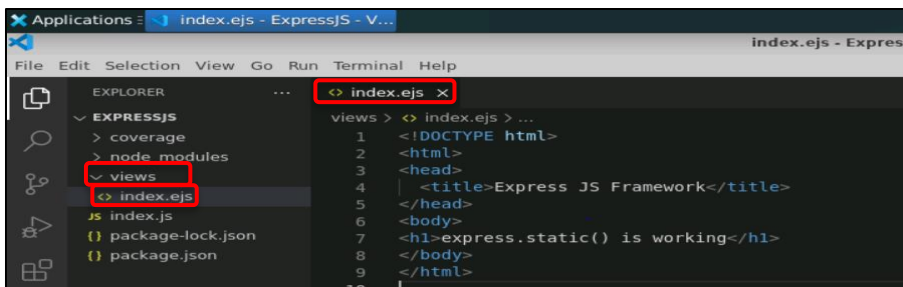
```

- 4.3 Create a **views** folder and then create an **index.ejs** file, and add the following code to it:

```

<!DOCTYPE html>
<html>
<head>
  <title>Express JS Framework</title>
</head>
<body>
<h1>express.static() is working</h1>
</body>
</html>

```

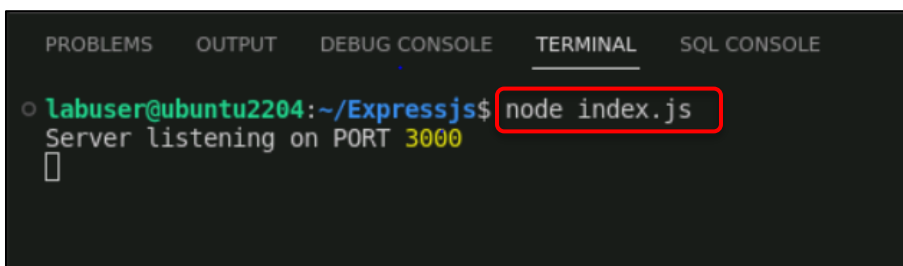


```

Applications  index.ejs - ExpressJS - V...
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPRESSJS
  coverage
  node_modules
  views
  index.ejs
  index.js
  package-lock.json
  package.json
views > index.ejs > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title>Express JS Framework</title>
5 </head>
6 <body>
7 <h1>express.static() is working</h1>
8 </body>
9 </html>

```

- 4.4 Run the **node index.js** command in the terminal



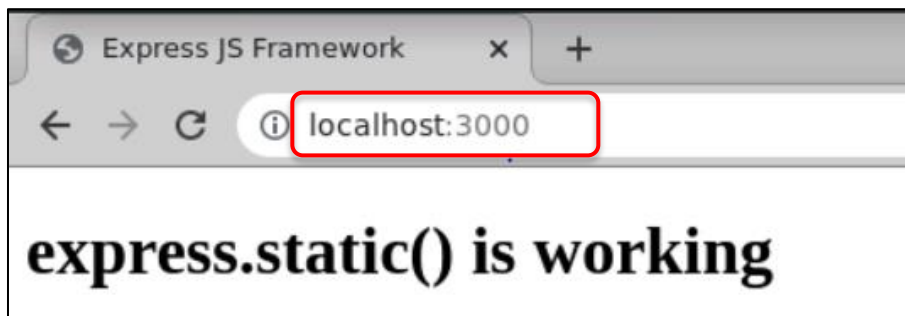
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000

```



4.5 Run the <http://localhost:3000> in the browser and check the output



## Step 5: Handle Error States in Express.js

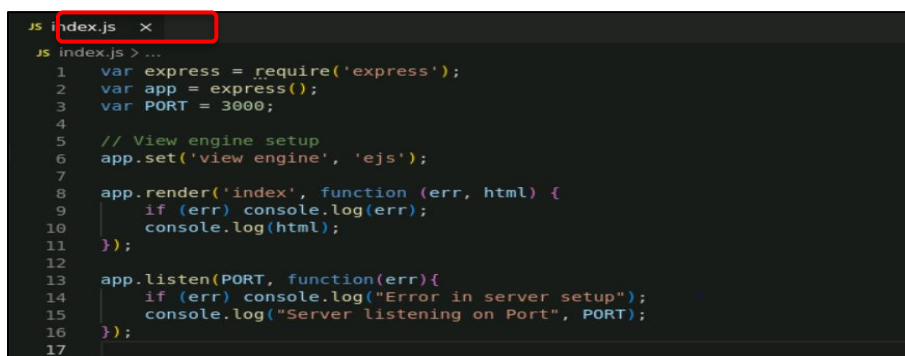
5.1 Write the following code in the `index.js` file:

```
var express = require('express');
var app = express();
var PORT = 3000;

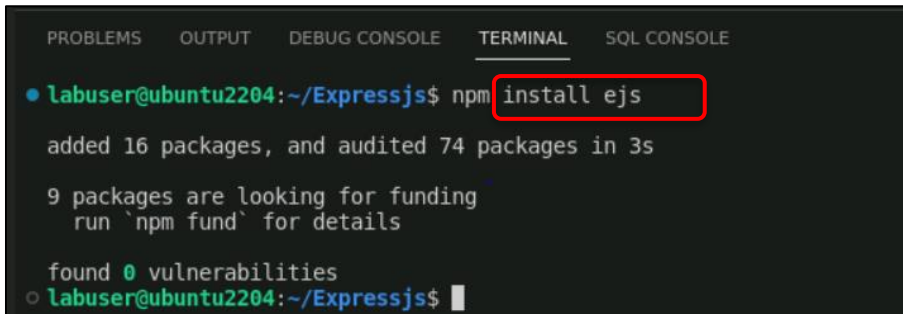
// View engine setup
app.set('view engine', 'ejs');

app.render('index', function (err, html) {
  if (err) console.log(err);
  console.log(html);
});

app.listen(PORT, function(err){
  if (err) console.log("Error in server setup");
  console.log("Server listening on Port", PORT);
});
```



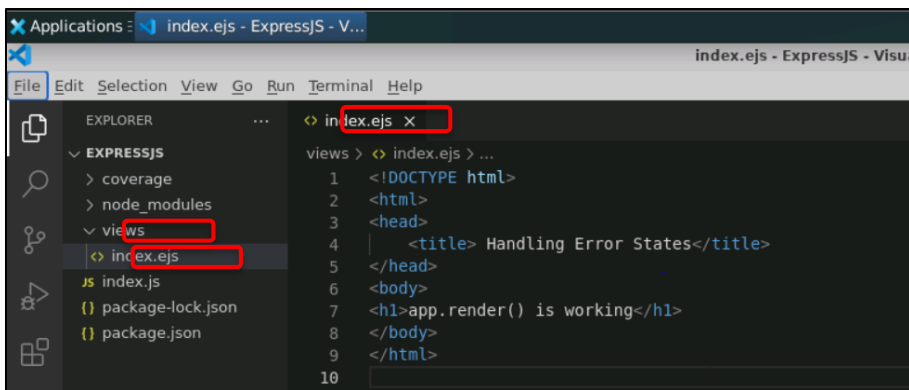
5.2 Run the below command in the terminal to add **ejs** to the project:  
**npm install ej**



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
• labuser@ubuntu2204:~/Expressjs$ npm install ej
added 16 packages, and audited 74 packages in 3s
9 packages are looking for funding
  run `npm fund` for details
found 0 vulnerabilities
• labuser@ubuntu2204:~/Expressjs$
```

5.3 Create a **views** folder. In that, create an **index.ejs** file and write the following code to it:

```
<!DOCTYPE html>
<html>
<head>
  <title> Handling Error States</title>
</head>
<body>
<h1>app.render() is working</h1>
</body>
</html>
```



```
Applications: index.ejs - ExpressJS - V...
index.ejs - ExpressJS - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPRESSJS
  coverage
  node_modules
  views
    index.ejs
  index.js
  package-lock.json
  package.json
views > index.ejs > ...
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title> Handling Error States</title>
5 </head>
6 <body>
7 <h1>app.render() is working</h1>
8 </body>
9 </html>
10
```

#### 5.4 Run the **node index.js** command in the terminal



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ node index.js
<!DOCTYPE html>
<html>
<body>
<h1>Handling Error States</h1>
</body>
</html>

Server listening on Port 3000
```

### Step 6: Debug in Express.js Module:

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

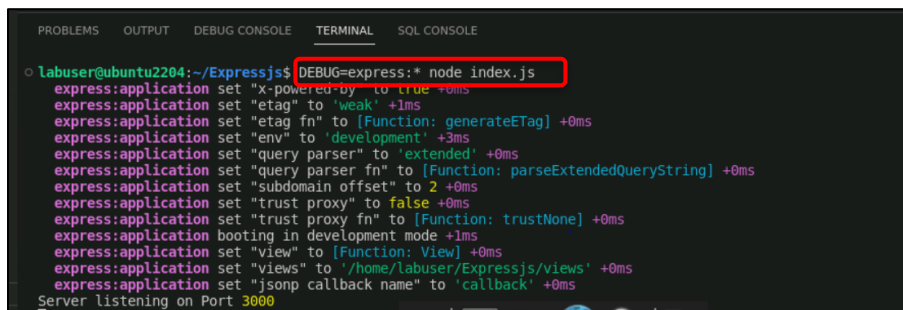
```
var express = require('express');
var app = express();
var PORT = 3000;

app.listen(PORT, function(err){
  if (err) console.log("Error in server setup");
  console.log("Server listening on Port", PORT);
});
```



```
JS index.js X
JS index.js > ...
1  var express = require('express');
2  var app = express();
3  var PORT = 3000;
4
5  app.listen(PORT, function(err){
6    if (err) console.log("Error in server setup");
7    console.log("Server listening on Port", PORT);
8  });
9
```

## 6.2 Run the **DEBUG=express:\* node index.js** command in the terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
labuser@ubuntu2204:~/Expressjs$ DEBUG=express:* node index.js
express:application set "x-powered-by" to true +0ms
express:application set "etag" to 'weak' +1ms
express:application set "etag fn" to [Function: generateETag] +0ms
express:application set "env" to 'development' +3ms
express:application set "query parser" to 'extended' +0ms
express:application set "query parser fn" to [Function: parseExtendedQueryString] +0ms
express:application set "subdomain offset" to 2 +0ms
express:application set "trust proxy" to false +0ms
express:application set "trust proxy fn" to [Function: trustNone] +0ms
express:application booting in development mode +1ms
express:application set "view" to [Function: View] +0ms
express:application set "views" to '/home/labuser/Expressjs/views' +0ms
express:application set "jsonp callback name" to 'callback' +0ms
Server listening on Port 3000
```

By following these steps, you have successfully implemented key features of the Express.js framework to enhance the proficiency in web development