

Jessa Marie B. Sanchez	
CITE 006-IT32S1	

Laboratory 6 - Node.js - NPM

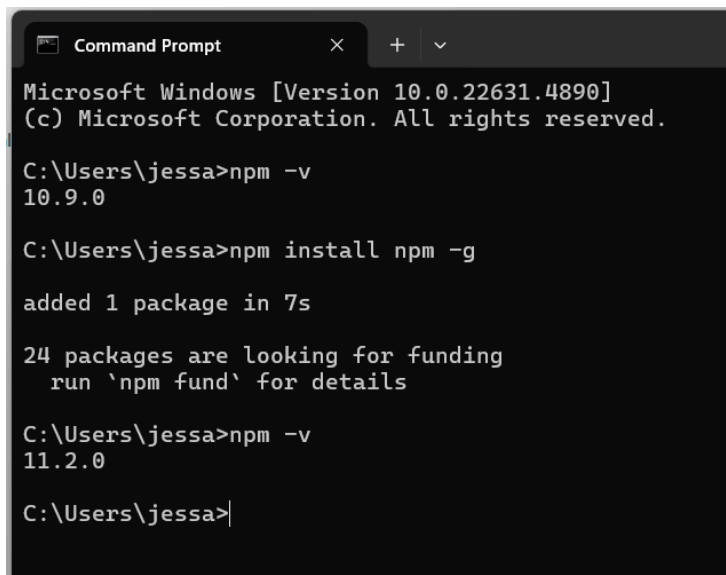
Objective: Familiarize students with NPM, its usage, and its importance in Node.js development.

Prerequisites:

- Basic knowledge of JavaScript and Node.js
- Node.js installed on each student's computer

Step 1: Installing NPM

1. Ensure Node.js is installed on the computer. NPM is automatically installed with Node.js.
2. Verify NPM installation by running `npm -v` in the terminal.



```
Command Prompt
Microsoft Windows [Version 10.0.22631.4890]
(c) Microsoft Corporation. All rights reserved.

C:\Users\jessa>npm -v
10.9.0

C:\Users\jessa>npm install npm -g
added 1 package in 7s
24 packages are looking for funding
  run 'npm fund' for details

C:\Users\jessa>npm -v
11.2.0

C:\Users\jessa>
```

Step 2: Creating a New Project

1. Create a new directory for the project.
2. Open a terminal and navigate to the project directory.
3. Initialize a new Node.js project by running `npm init` and following the prompts to set up the project details. This brings you to the initialization utility.



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal output shows the following commands and results:

```
PS C:\Users\jessa> mkdir NPM
```

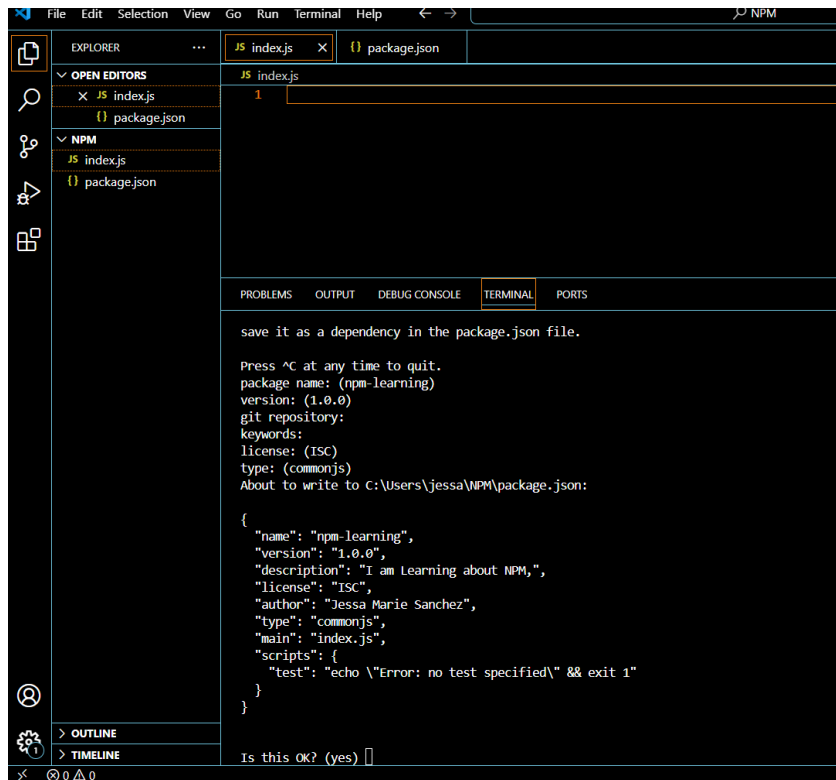
Directory: C:\Users\jessa

Mode	LastWriteTime	Length	Name
d----	09/03/2025 5:00 am		NPM

```
PS C:\Users\jessa> cd NPM
PS C:\Users\jessa\NPM> code .
PS C:\Users\jessa\NPM> 
```

Step 3: Installing Packages

1. The concept of packages and dependencies.
2. Demonstrate how to install packages using NPM.



The screenshot shows the Visual Studio Code interface with the Explorer, Search, Source Control, and Run and Debug views on the left. The Explorer view shows the 'index.js' and 'package.json' files. The Search view shows the 'index.js' file. The Source Control view shows the 'index.js' file. The Run and Debug view shows the 'index.js' file. The terminal window at the bottom displays the output of the 'npm init' command, which prompts the user to enter package name, version, git repository, keywords, license, type, and about. The user has entered 'npm-learning', '1.0.0', and 'I am Learning about NPM,'. The terminal shows the resulting 'package.json' file content.

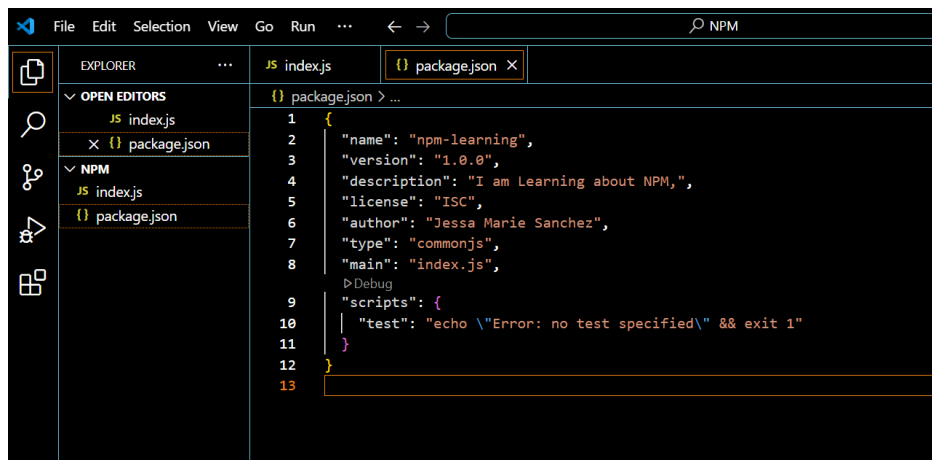
```
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (npm-learning)
version: (1.0.0)
git repository:
keywords:
license: (ISC)
type: (commonjs)
About to write to C:\Users\jessa\NPM\package.json:

{
  "name": "npm-learning",
  "version": "1.0.0",
  "description": "I am Learning about NPM,",
  "license": "ISC",
  "author": "Jessa Marie Sanchez",
  "type": "commonjs",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}

Is this OK? (yes) ☐
```

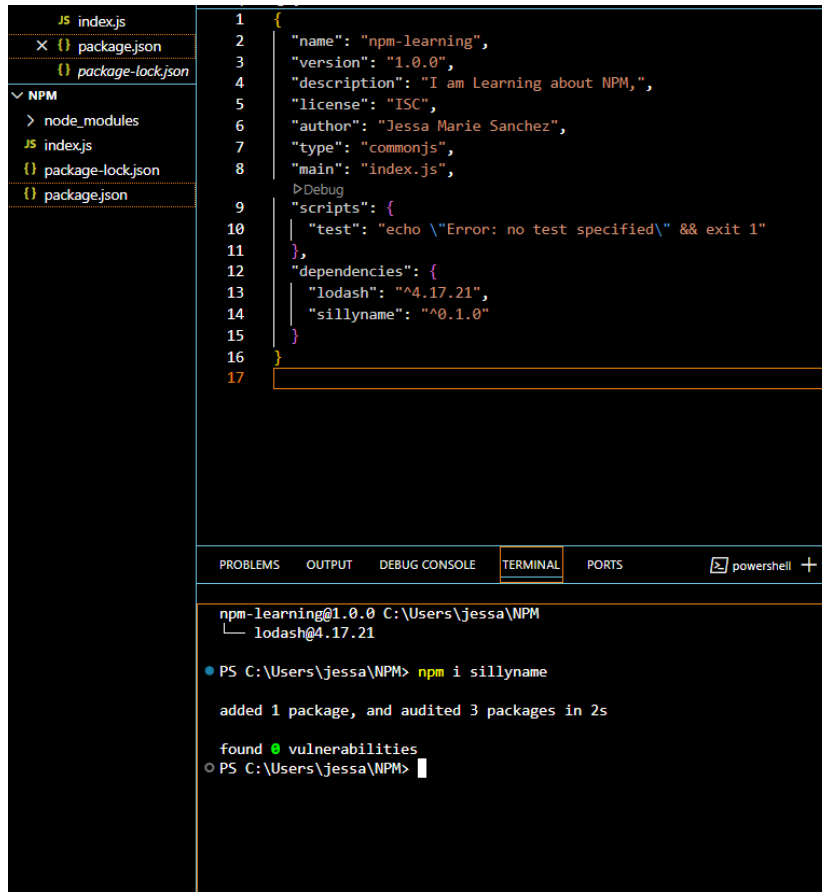
When type y. It will create another file that contains the configuration file of the project.



The screenshot shows the Visual Studio Code interface with the Explorer, Search, Source Control, and Run and Debug views on the left. The Explorer view shows the 'index.js' and 'package.json' files. The Search view shows the 'index.js' file. The Source Control view shows the 'index.js' file. The Run and Debug view shows the 'index.js' file. The terminal window at the bottom displays the content of the 'package.json' file, which is a JSON object containing project metadata and scripts.

```
{
  "name": "npm-learning",
  "version": "1.0.0",
  "description": "I am Learning about NPM,",
  "license": "ISC",
  "author": "Jessa Marie Sanchez",
  "type": "commonjs",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  }
}
```

1. Install a few popular packages such as `lodash` or `axios` using `npm install <package-name>`.
2. Verify that the packages are installed by checking the `node_modules` directory and `package.json` file.



The screenshot shows the VS Code interface. On the left, the Explorer sidebar shows the file structure: `index.js`, `package.json`, `package-lock.json`, and an `NPM` folder containing `node_modules`, `index.js`, `package-lock.json`, and `package.json`. The main editor displays the `package.json` file with the following content:

```
1 {
2   "name": "npm-learning",
3   "version": "1.0.0",
4   "description": "I am Learning about NPM,",
5   "license": "ISC",
6   "author": "Jessa Marie Sanchez",
7   "type": "commonjs",
8   "main": "index.js",
9   >Debug
10  "scripts": {
11    | "test": "echo \"Error: no test specified\" && exit 1"
12  },
13  "dependencies": {
14    | "lodash": "^4.17.21",
15    | "sillyname": "^0.1.0"
16  }
17 }
```

Below the editor, the TERMINAL tab is active, showing the command prompt output:

```
npm-learning@1.0.0 C:\Users\jessa\NPM
└─┬─ lodash@4.17.21
   │
   └─ PS C:\Users\jessa\NPM> npm i sillyname

added 1 package, and audited 3 packages in 2s

found 0 vulnerabilities
○ PS C:\Users\jessa\NPM> |
```

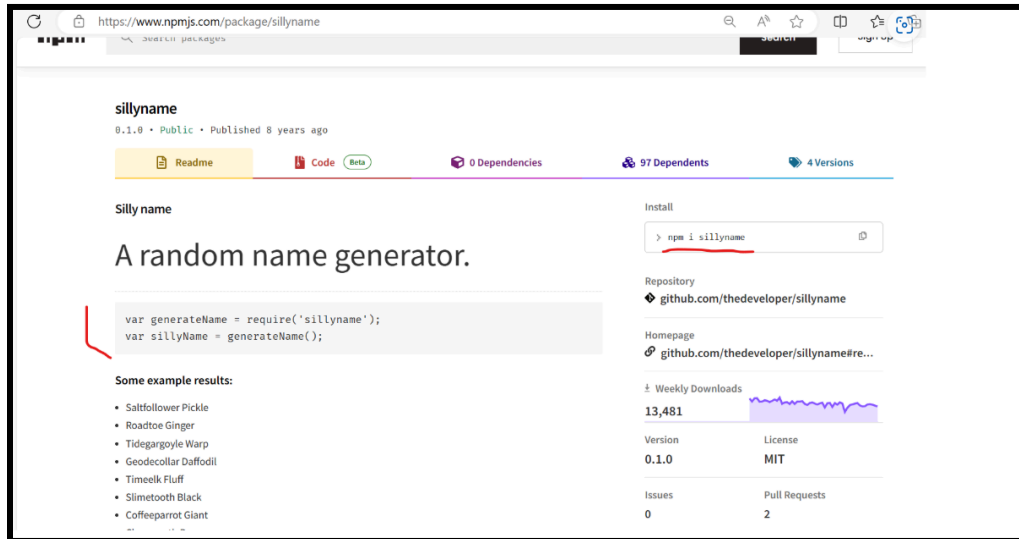
4. Open the `index.js` and type the code below.



The screenshot shows the VS Code interface with the `index.js` file open in the editor. The Explorer sidebar on the left shows the file structure: `index.js`, `package.json`, `package-lock.json`, and an `NPM` folder containing `node_modules`, `index.js`, `package-lock.json`, and `package.json`. The main editor displays the `index.js` file with the following code:

```
1 var generateName = require('sillyname');
2 var sillyname = generateName();
```

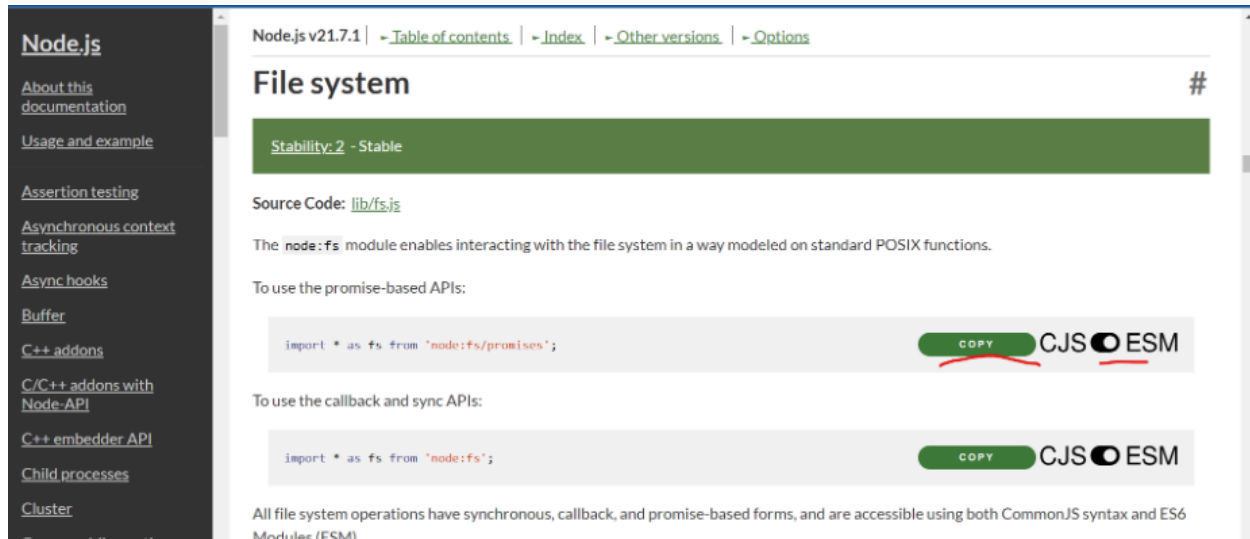
This is came from the package in the npm site.



Add the following next line of code . Then run in the terminal using node index command.

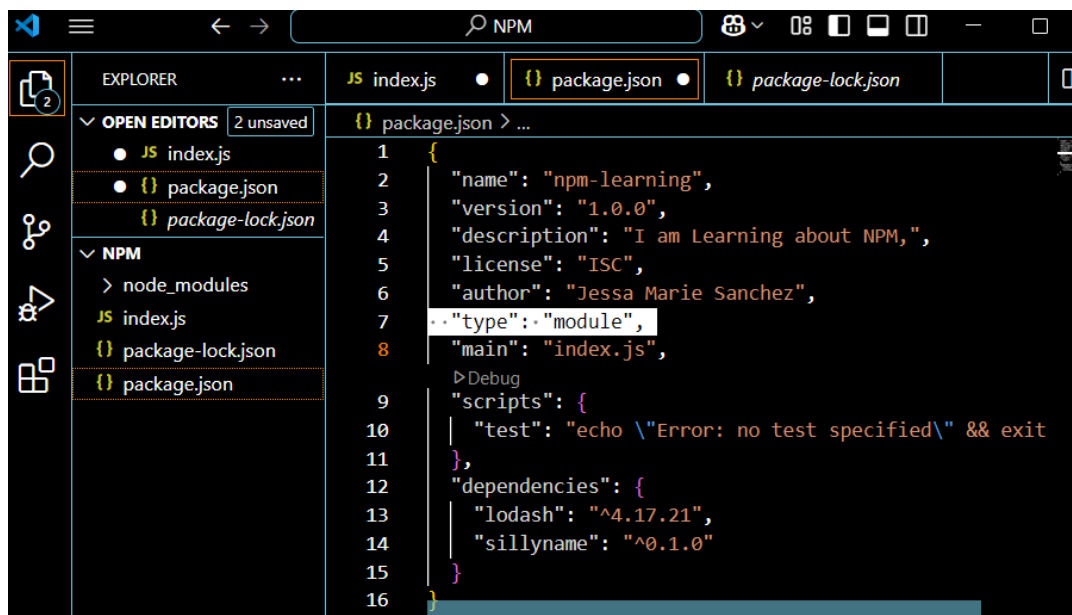


ECMAScript serves as the foundation for JavaScript development, providing a standardized specification that evolves over time to meet the needs of web developers and ensure the interoperability and compatibility of JavaScript implementations across different platforms and environments.

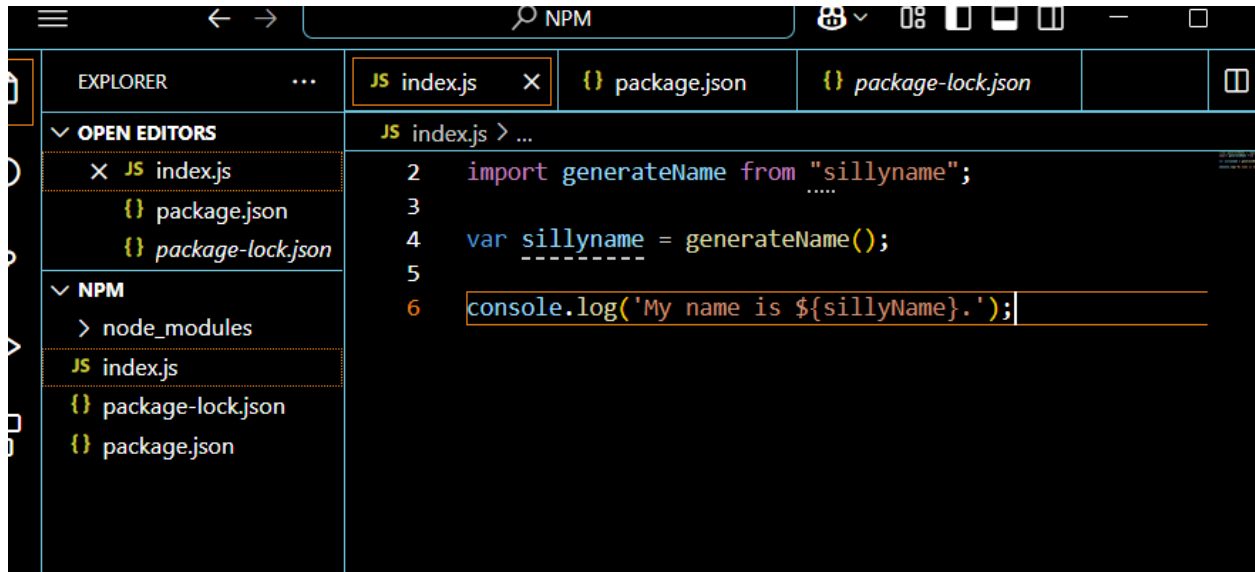


1. Go to package.json and add type then with a value module to use the ECMA script.

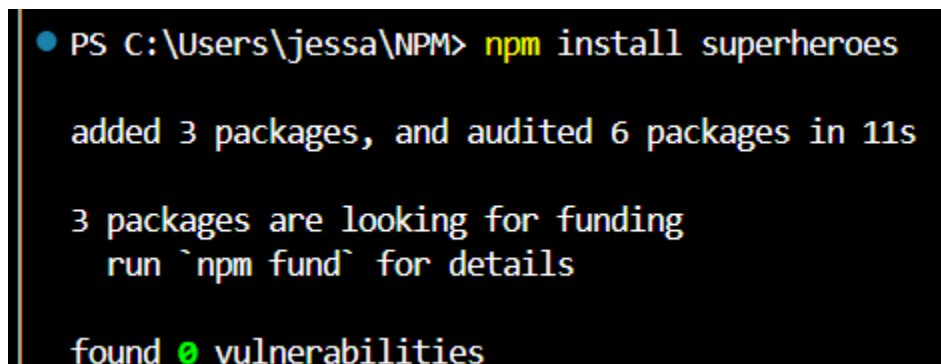
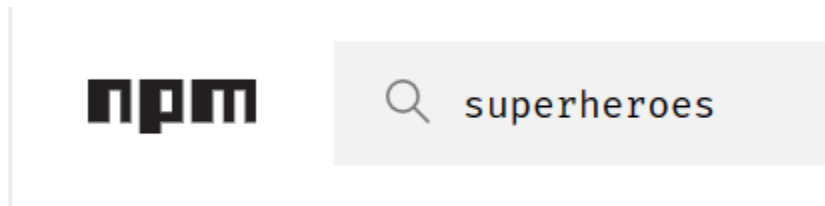
//On this part from **"type": "commonjs"** we change it into **"type": "module"**,



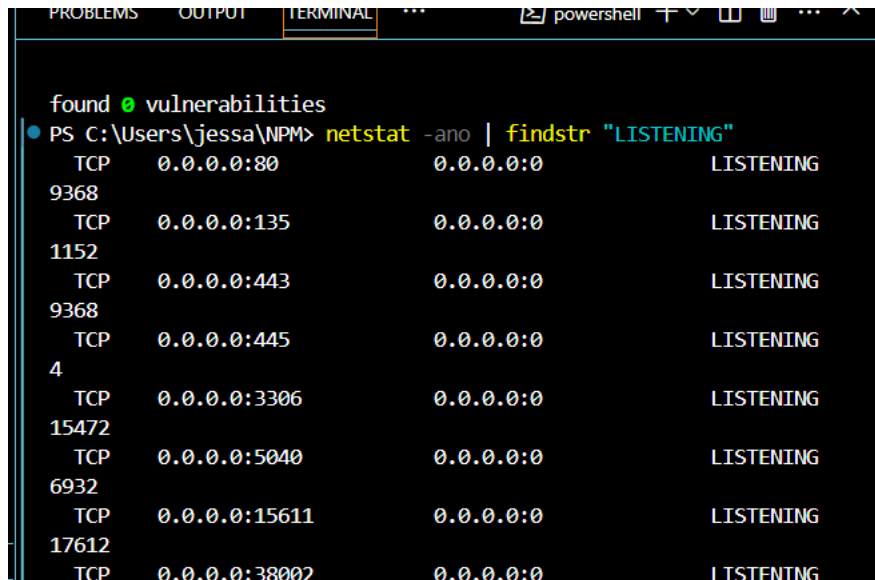
2. Modify your code in index.js. Run it with node index.



Task 1: Use the ECMA script version of code, create a program that generate various name of superheroes. Hint: look the package superheroes.

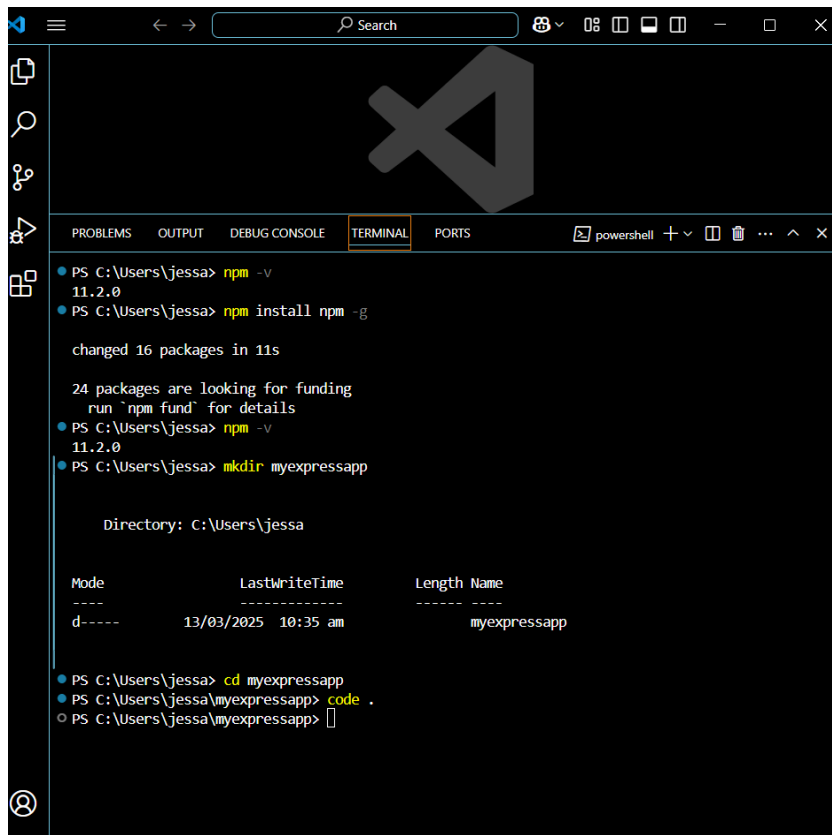


-ano | findstr "LISTENING"



```
found 0 vulnerabilities
PS C:\Users\jessa\NPM> netstat -ano | findstr "LISTENING"
TCP    0.0.0.0:80          0.0.0.0:0        LISTENING
9368
TCP    0.0.0.0:135        0.0.0.0:0        LISTENING
1152
TCP    0.0.0.0:443        0.0.0.0:0        LISTENING
9368
TCP    0.0.0.0:445        0.0.0.0:0        LISTENING
4
TCP    0.0.0.0:3306       0.0.0.0:0        LISTENING
15472
TCP    0.0.0.0:5040       0.0.0.0:0        LISTENING
6932
TCP    0.0.0.0:15611      0.0.0.0:0        LISTENING
17612
TCP    0.0.0.0:38002      0.0.0.0:0        LISTENING
```

PART 2



```
PS C:\Users\jessa> npm -v
11.2.0
PS C:\Users\jessa> npm install npm -g

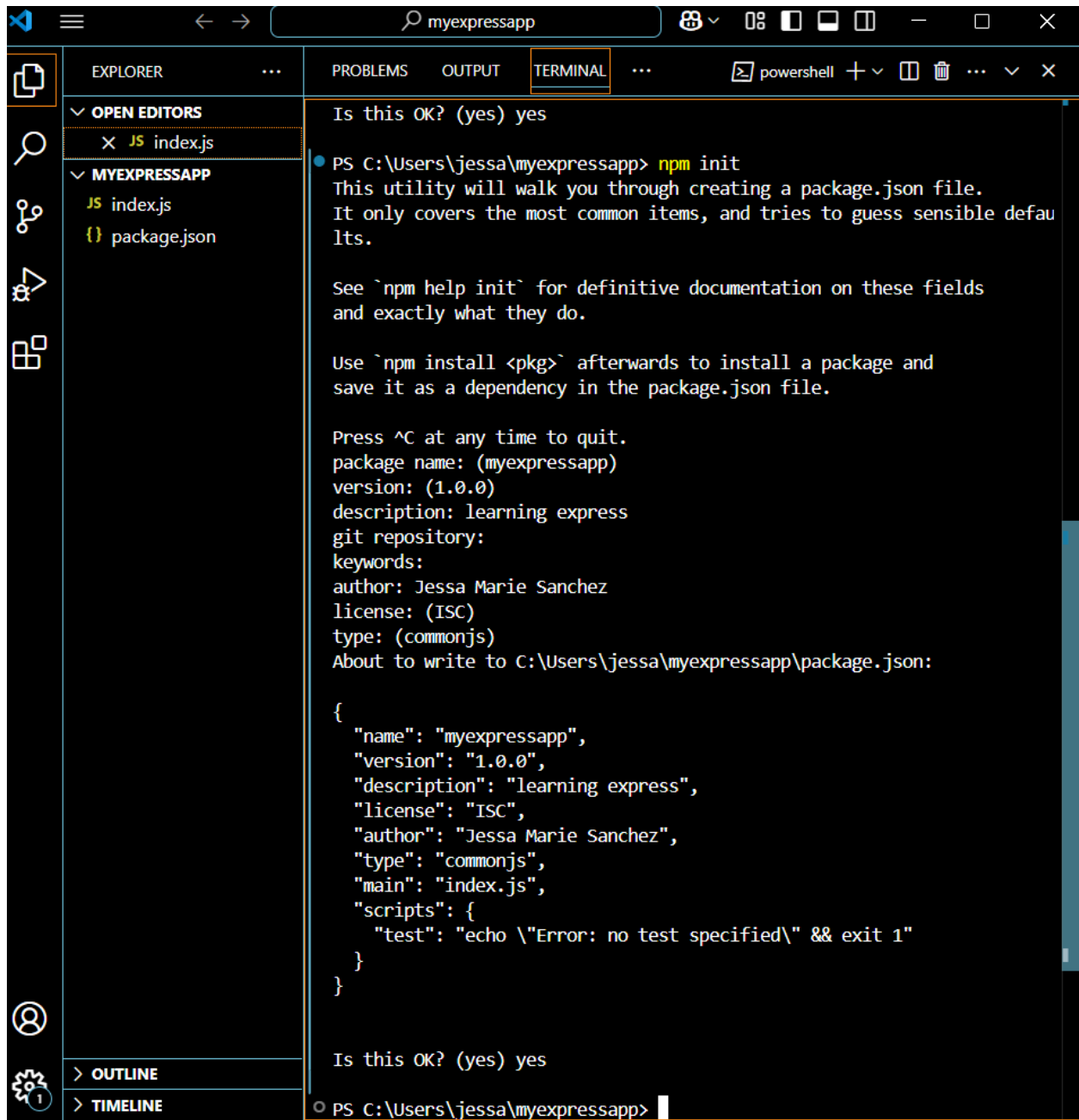
changed 16 packages in 11s

24 packages are looking for funding
  run `npm fund` for details
PS C:\Users\jessa> npm -v
11.2.0
PS C:\Users\jessa> mkdir myexpressapp

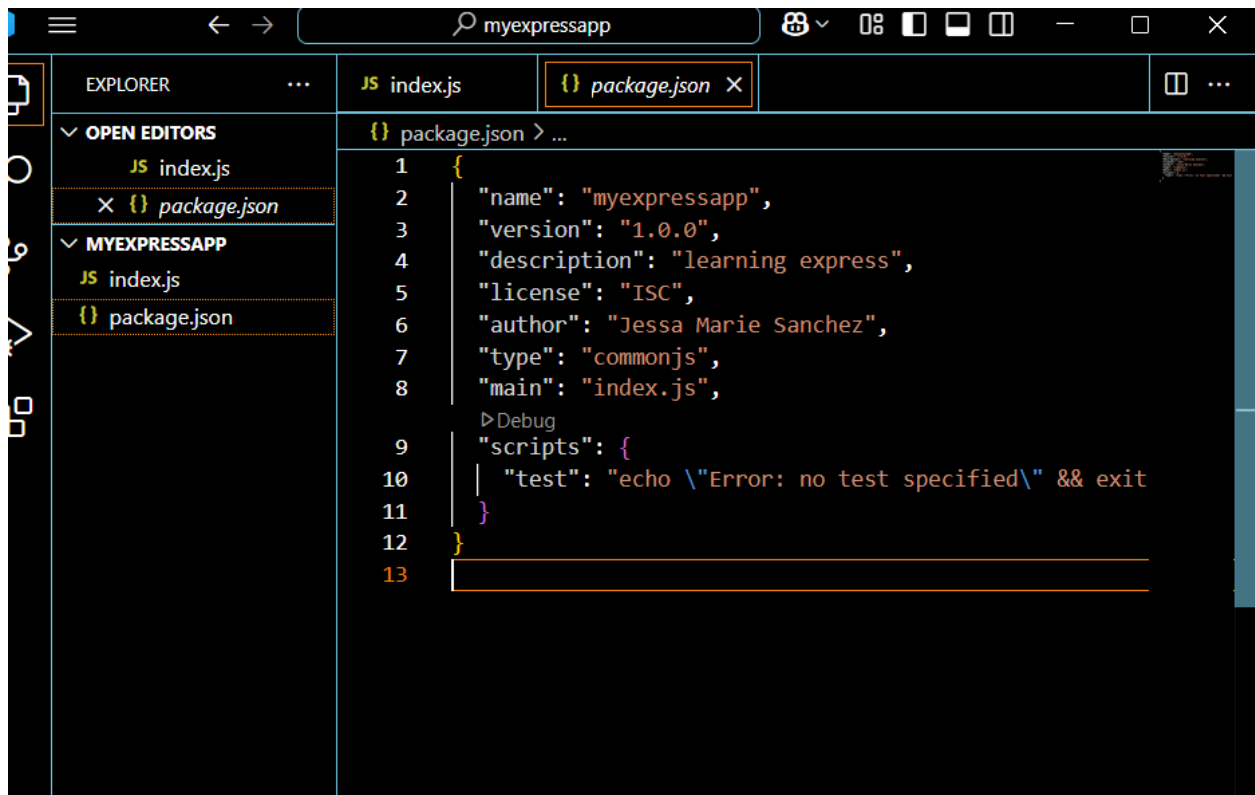
Directory: C:\Users\jessa

Mode                LastWriteTime         Length Name
----                -
d-----          13/03/2025  10:35 am             myexpressapp

PS C:\Users\jessa> cd myexpressapp
PS C:\Users\jessa\myexpressapp> code .
PS C:\Users\jessa\myexpressapp>
```

One click yes it will create package.json that shows the created details in our project. (Below)



You can also initialize using `npm init -y` is a command used to initialize a new Node.js project with default settings without prompting for input.

4. Install the Express package

Now install Express in the `myexpressapp` directory and save it in the dependencies list.

myexpressapp

EXPLORER

JS index.js

package-lock.json

OPEN EDITORS

JS index.js

package-lock.json

MYEXPRESSAPP

node_modules

JS index.js

package-lock.json

package.json

package-lock.json > ...

1 {

2 "name": "myexpressapp",

3 "version": "1.0.0",

4 "lockfileVersion": 3,

5 "requires": true,

6 "packages": {

7 "": {

8 "name": "myexpressapp",

9 "version": "1.0.0",

10 "license": "ISC",

11 "dependencies": {

12 | "express": "^4.21.2"

13 | }

14 },

15 "node_modules/accepts": {

16 "version": "1.3.8",

17 "resolved": "https://registry.npmjs.org/accepts/",

18 "integrity": "sha512-PYAthTa2m2VKxuvSD3DPC/Gy+U+

19 "license": "MIT",

20 "dependencies": {

PROBLEMS

OUTPUT

TERMINAL

powerShell

}

PS C:\Users\jessa\myexpressapp> npm install express

added 69 packages, and audited 70 packages in 12s

14 packages are looking for funding

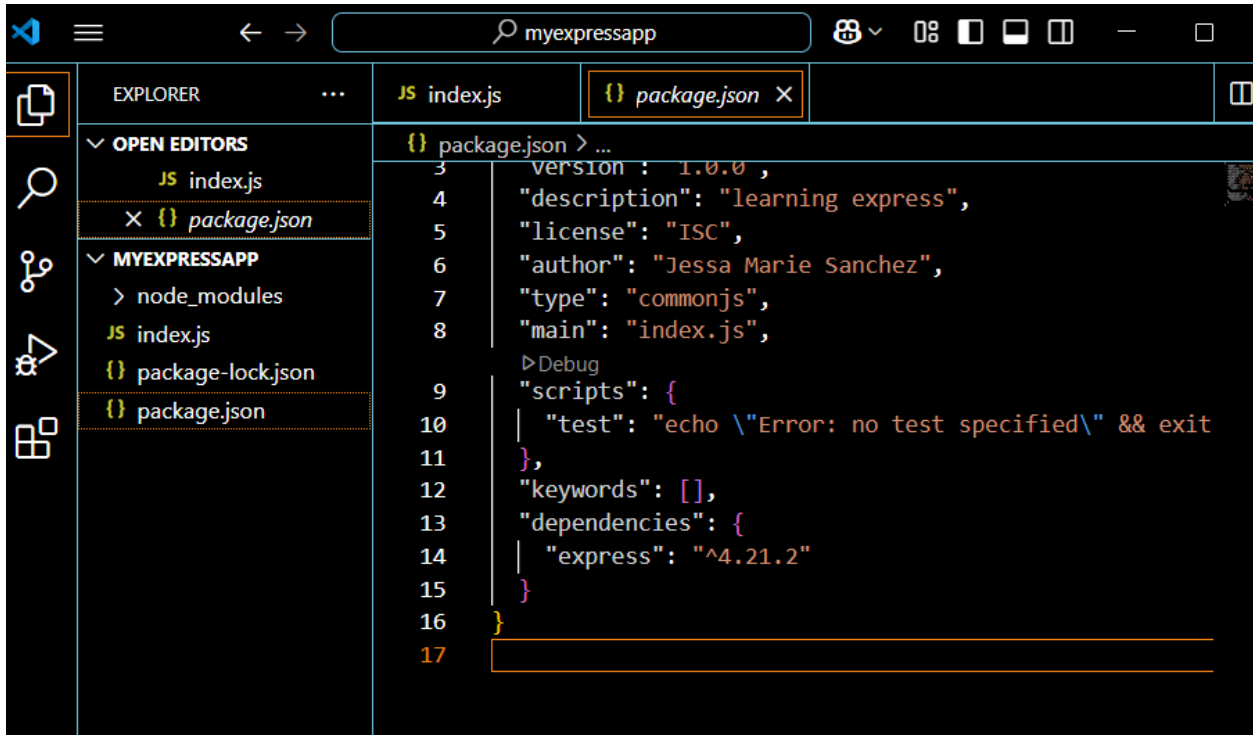
run `npm fund` for details

found 0 vulnerabilities

PS C:\Users\jessa\myexpressapp>

OUTLINE

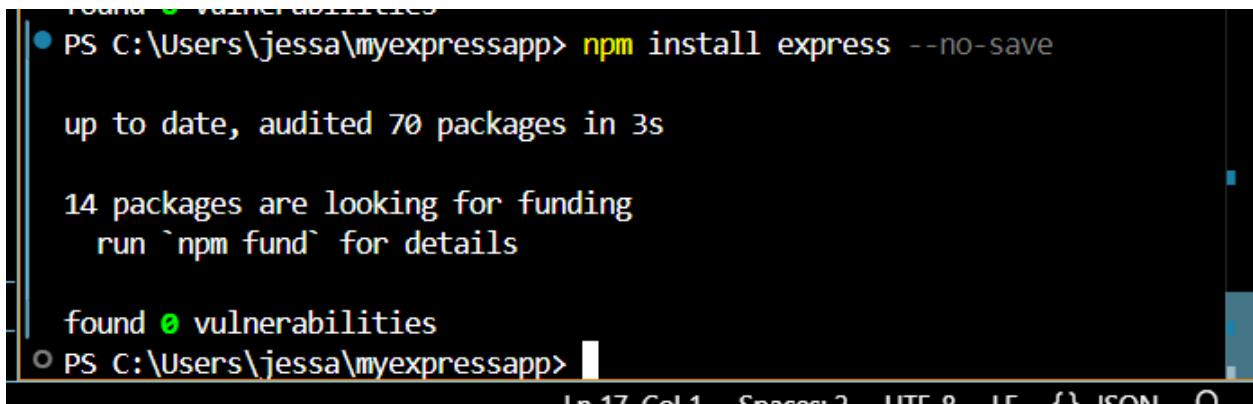
TIMELINE



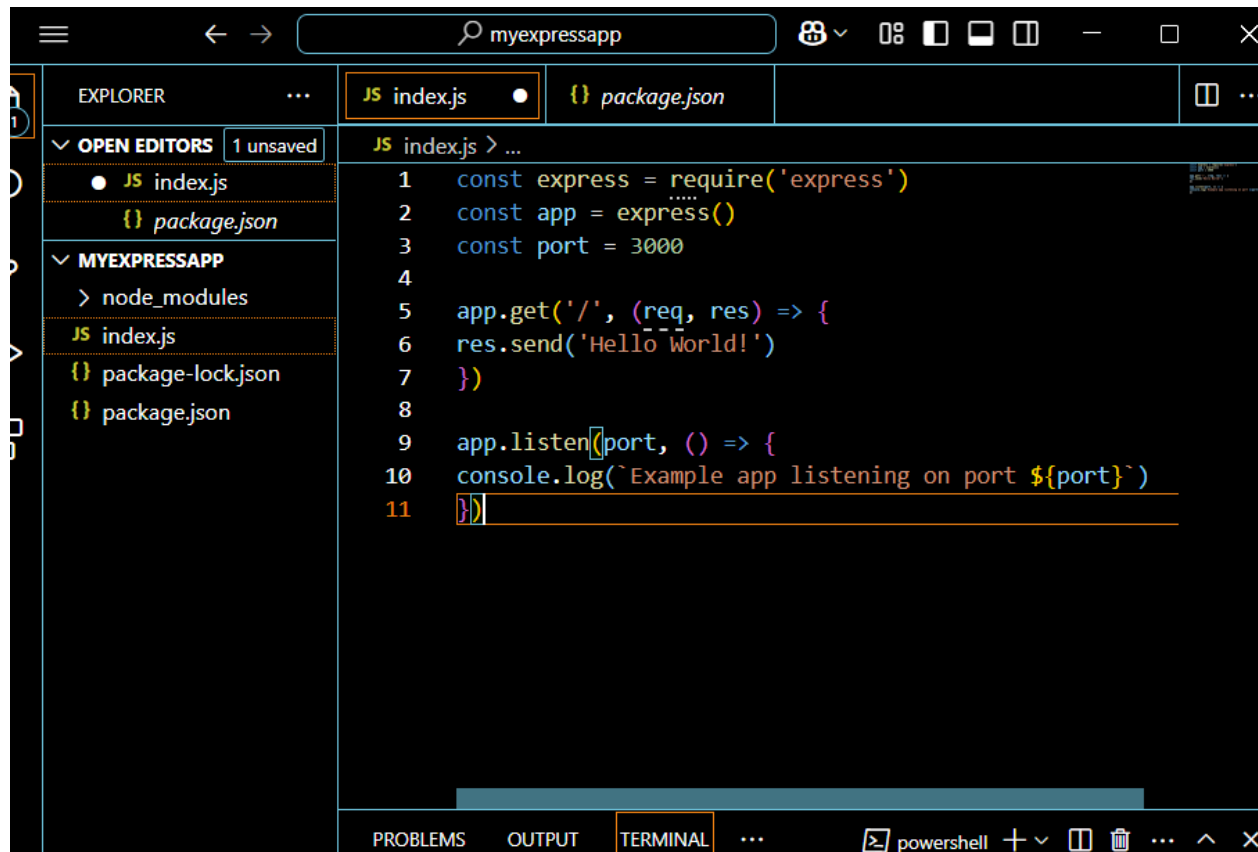
To install Express temporarily and not add it to the dependencies list:

Unset

```
npm install express --no-save
```



5. Write Server application in index.js

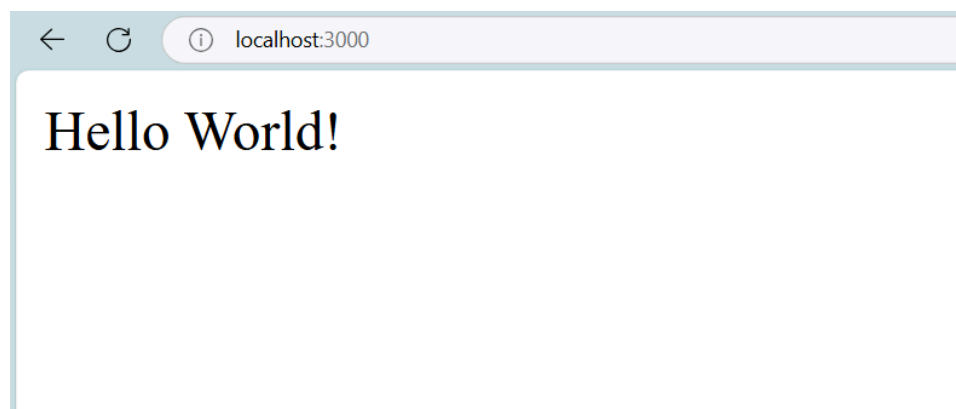


The screenshot shows the Visual Studio Code editor interface. The Explorer sidebar on the left shows the project structure with 'index.js' selected. The main editor area displays the following JavaScript code in 'index.js':

```
1 const express = require('express')
2 const app = express()
3 const port = 3000
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!')
7 })
8
9 app.listen(port, () => {
10  console.log(`Example app listening on port ${port}`)
11 })
```

6. Start Server using the command below and access the localhost:3000 to find the output.

node index



Task2:

a. Compile your output

b. Access this website

[https://nodejs.org/docs/latest/api/fs.html#file-system.](https://nodejs.org/docs/latest/api/fs.html#file-system)

[Links to an external site.](#)

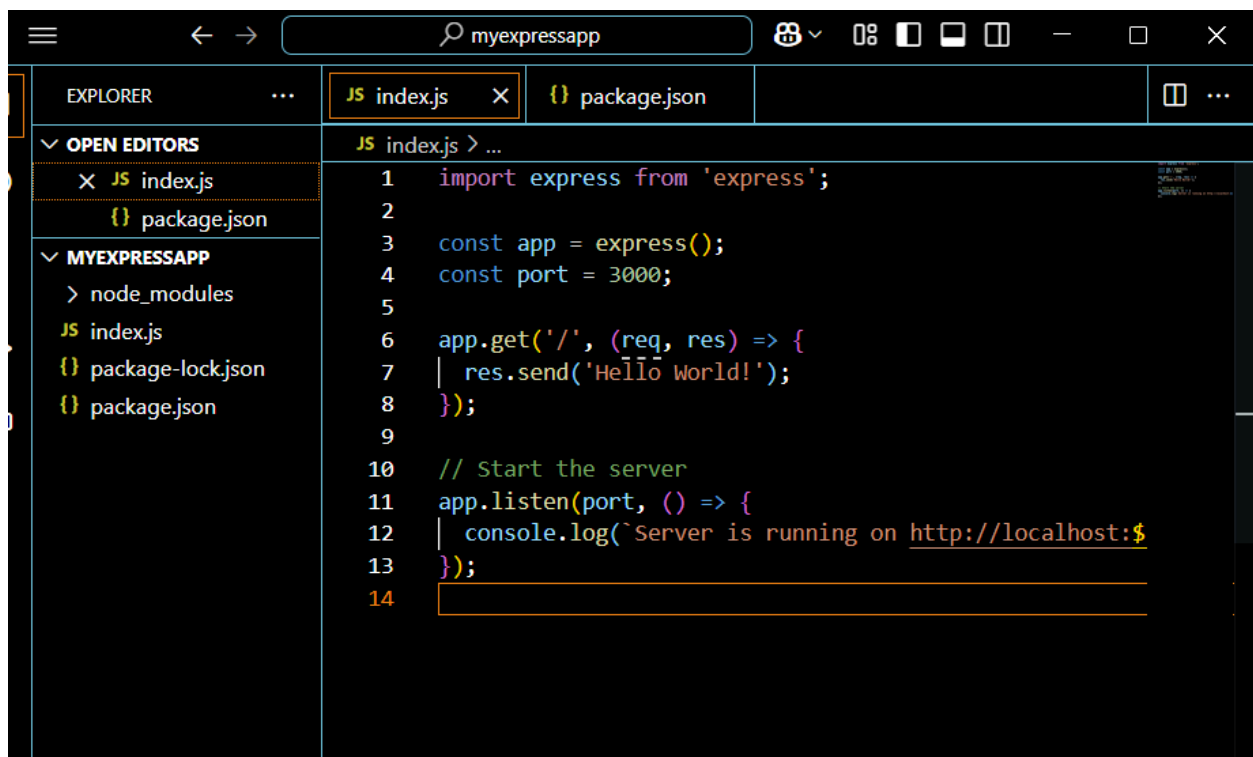
Analyze its content then modify our index.js to make it ESM format.

c. Add the following code in the index.js.

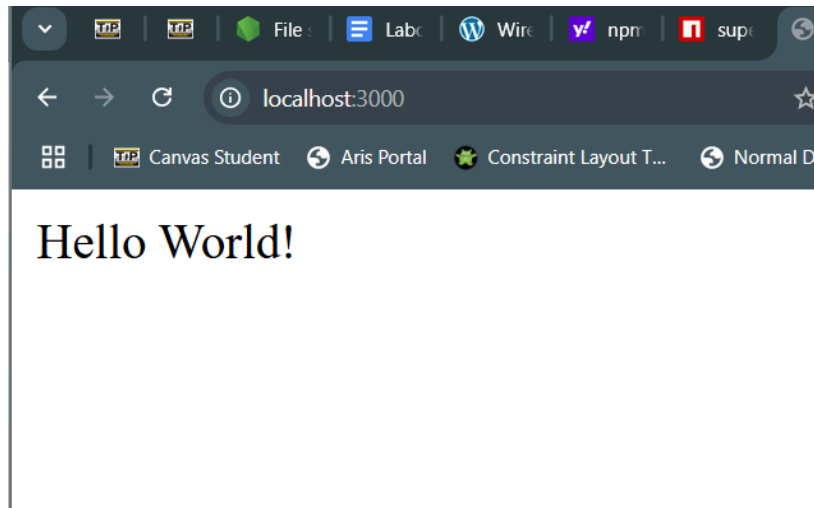
// Start the server

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

c. Compile your modified **code** and the output in the **console** and **server**.



```
PS C:\Users\jessa\myexpressapp> node index.js
Server is running on http://localhost:3000
```



Part 3: Express Application Generator

1. Access this page for more detailed steps.
<https://expressjs.com/en/starter/generator.html>

Express

search

HomeGetting startedGuideAPI referenceAdvanced topicsResourcesSupportBlog

English

Express application generator

Use the application generator tool, `express-generator`, to quickly create an application skeleton. You can run the application generator with the `npm` command (available in Node.js 8.2.0).

```
$ npm install -g express-generator
$ express
```

For earlier Node versions, install the application generator as a global npm package and then launch it:

```
$ npm install -g express-generator
$ express
```

Display the command options with the `-h` option:

```
$ express -h

Usage: express [options] [dir]

Options:
  -h, --help            output usage information
  --version              output the version number
  -e, --ejs              add ejs engine support
  -hbs                  add handlebars engine support
  -pug                  add pug engine support
  -H, --hogan            add hogan.js engine support
  --no-view              generate without view engine
  -v, --view <engine>   add view <engine> support (ejs|hbs|hjs|jade|pug|twig|vash) (defaults to jade)
  -c, --css <engine>    add stylesheet <engine> support (less|stylus|compass|sass) (defaults to plain css)
  --git                 add .gitignore
  -f, --force            force on non-empty directory
```

For example, the following creates an Express app named **myapp**. The app will be created in a folder named **myapp** in the current working directory and the view engine will be set to **pug**.

Express

search

HomeGetting startedGuideAPI referenceAdvanced topicsResourcesSupportBlog

English

```
$ express --view=pug myapp

create : myapp
create : myapp/package.json
create : myapp/app.js
create : myapp/public
create : myapp/public/javascripts
create : myapp/public/images
create : myapp/routes
create : myapp/routes/index.js
create : myapp/routes/users.js
create : myapp/public/stylesheets
create : myapp/public/stylesheets/style.css
create : myapp/views
create : myapp/views/index.pug
create : myapp/views/layout.pug
create : myapp/views/error.pug
create : myapp/bin
create : myapp/bin/www
```

Then install dependencies:

```
$ cd myapp
$ npm install
```

On MacOS or Linux, run the app with this command:

```
$ DEBUG=myapp:* npm start
```

On Windows Command Prompt, use this command:

```
> set DEBUG=myapp:* & npm start
```

← → ↺

expressjs.com/en/starter/generator.html

🔍 ☆ 🔌 📄 🔄 📌

📄 All Bookmarks

📄 Canvas Student

📄 Aris Portal

📄 Constraint Layout T...

📄 Normal Distribution...

📄 Assessment Task 5...

Express

search

HomeGetting startedGuideAPI referenceAdvanced topicsResourcesSupportBlog

English

On MacOS or Linux, run the app with this command:

```
$ DEBUG=myapp:* npm start
```

On Windows Command Prompt, use this command:

```
> set DEBUG=myapp:* & npm start
```

On Windows PowerShell, use this command:

```
ps $env:DEBUG="myapp:*"; npm start
```

Then, load `http://localhost:3000/` in your browser to access the app.

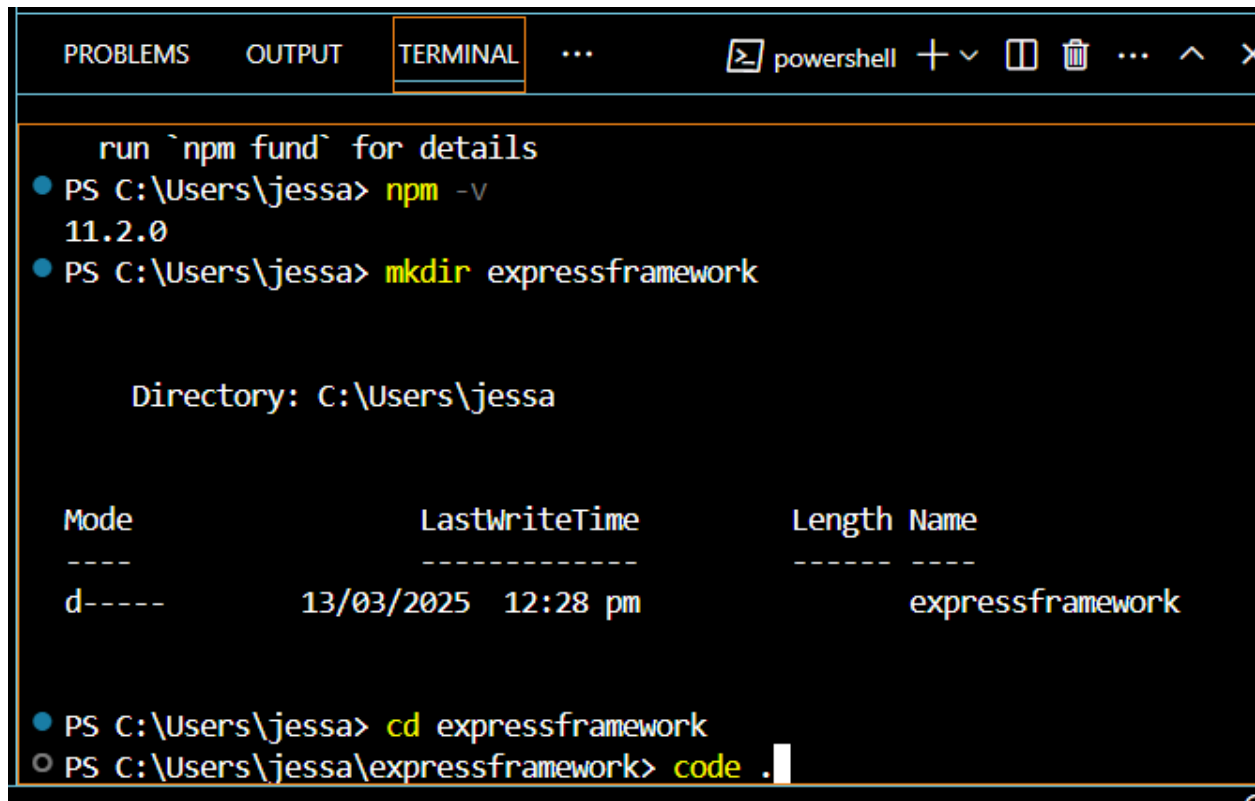
The generated app has the following directory structure:

```
+
├── app.js
├── bin
│   └── www
├── package.json
├── public
│   ├── images
│   ├── javascripts
│   └── stylesheets
│       └── style.css
├── routes
│   ├── index.js
│   └── users.js
├── views
│   ├── error.pug
│   ├── index.pug
│   └── layout.pug
└── 7 directories, 9 files
```

The app structure created by the generator is just one of many ways to structure Express apps. Feel free to use this structure or modify it to best suit your needs.

[Previous: Hello World](#) [Next: Basic routing](#)

2. Create another project. In your terminal, in my case I go back to htdocs



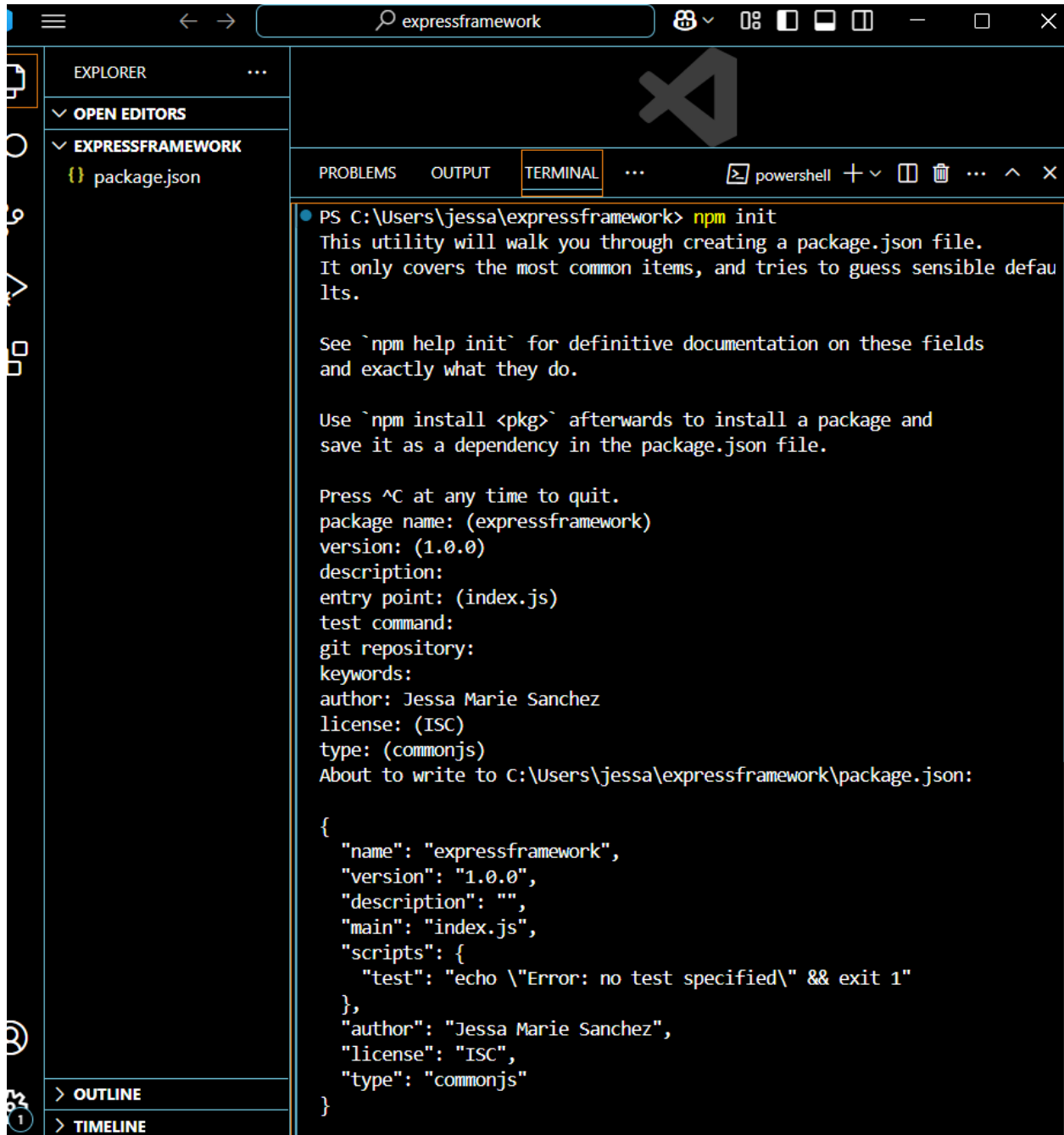
The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal is running a PowerShell session. The user has entered the command `npm -v` and received the output `11.2.0`. Then, they entered `mkdir expressframework` and the terminal shows the directory listing for `C:\Users\jessa`, which includes a new directory named `expressframework`. Finally, the user navigates into the new directory with `cd expressframework` and opens a new code editor window with `code .`.

```
run `npm fund` for details
• PS C:\Users\jessa> npm -v
11.2.0
• PS C:\Users\jessa> mkdir expressframework

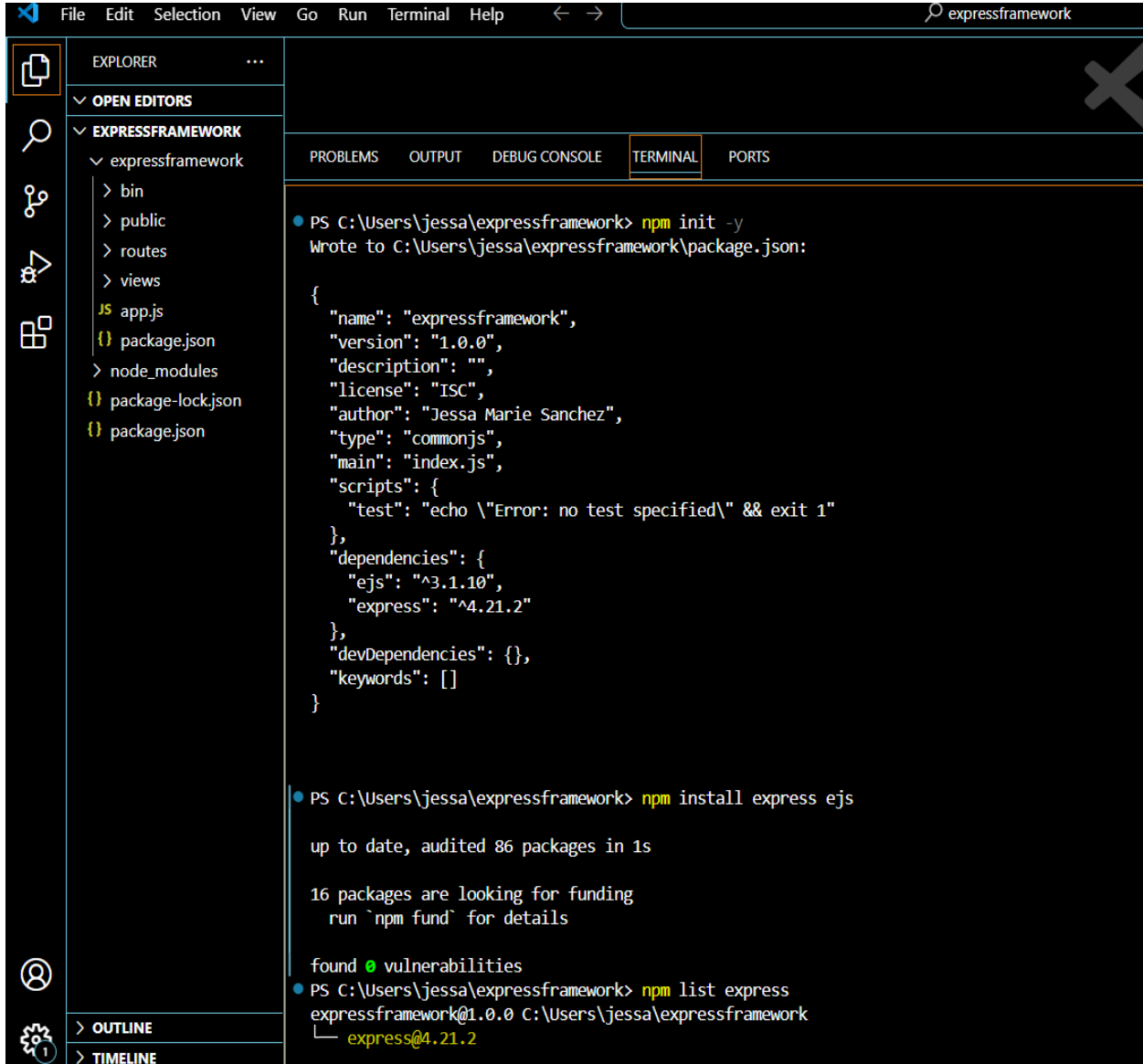
Directory: C:\Users\jessa

Mode                LastWriteTime         Length Name
----                -
d-----          13/03/2025  12:28 pm              expressframework

• PS C:\Users\jessa> cd expressframework
○ PS C:\Users\jessa\expressframework> code .
```



3. Type the following command



```
File Edit Selection View Go Run Terminal Help < -> expressframework
```

EXPLORER ...

OPEN EDITORS

EXPRESSFRAMEWORK

- expressframework
 - > bin
 - > public
 - > routes
 - > views
 - JS app.js
 - { package.json
 - > node_modules
 - { package-lock.json
 - { package.json

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\jessa\expressframework> npm init -y
Wrote to C:\Users\jessa\expressframework\package.json:

{
  "name": "expressframework",
  "version": "1.0.0",
  "description": "",
  "license": "ISC",
  "author": "Jessa Marie Sanchez",
  "type": "commonjs",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "dependencies": {
    "ejs": "^3.1.10",
    "express": "^4.21.2"
  },
  "devDependencies": {},
  "keywords": []
}

PS C:\Users\jessa\expressframework> npm install express ejs

up to date, audited 86 packages in 1s

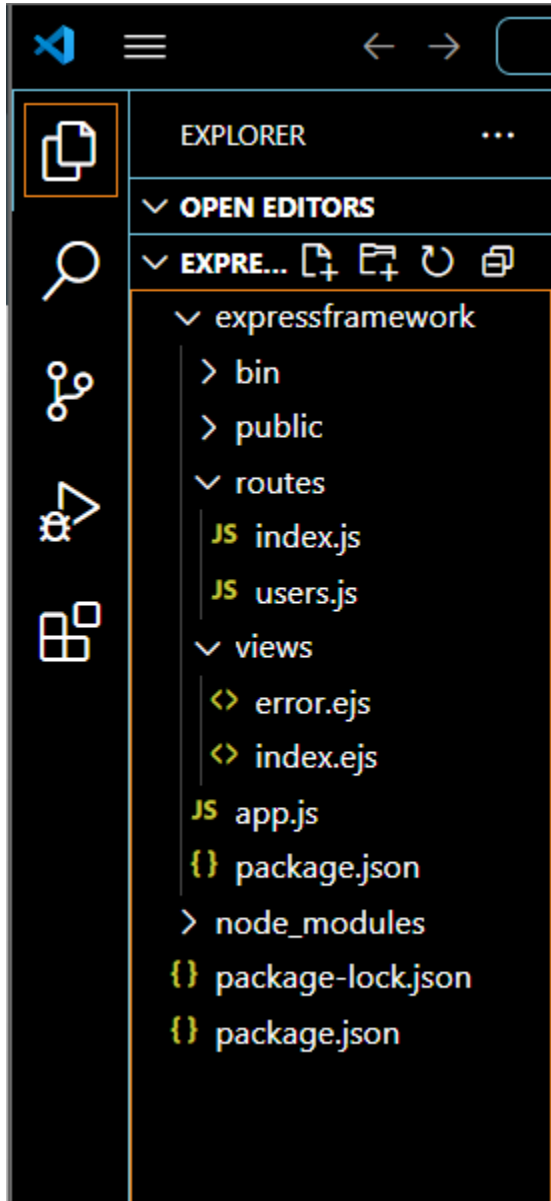
16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\jessa\expressframework> npm list express
expressframework@1.0.0 C:\Users\jessa\expressframework
├─ express@4.21.2
```

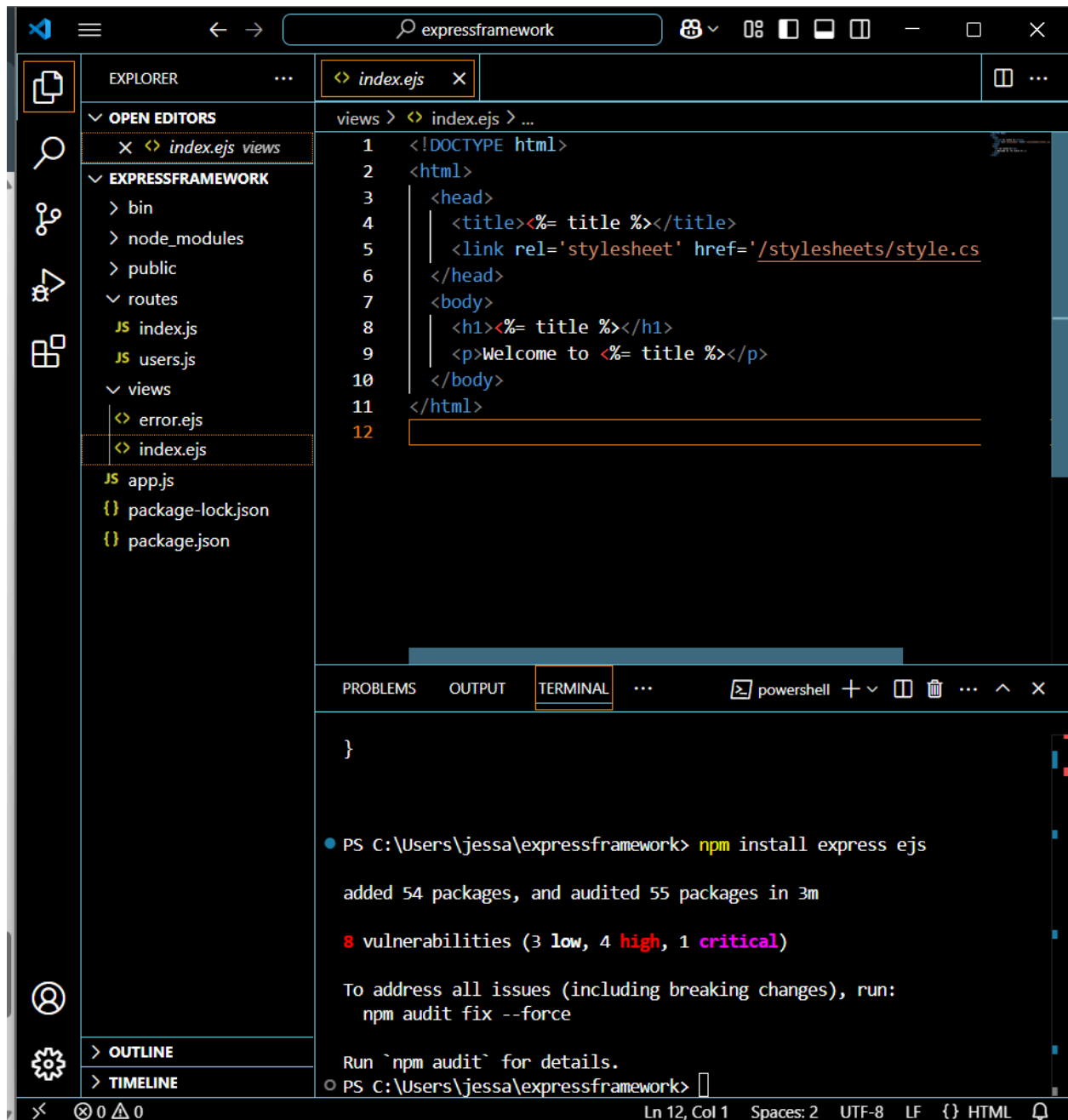
OUTLINE

TIMELINE

Automatically it will create view in the project explorer



4. In your VS code go to that directory expressframework using cd

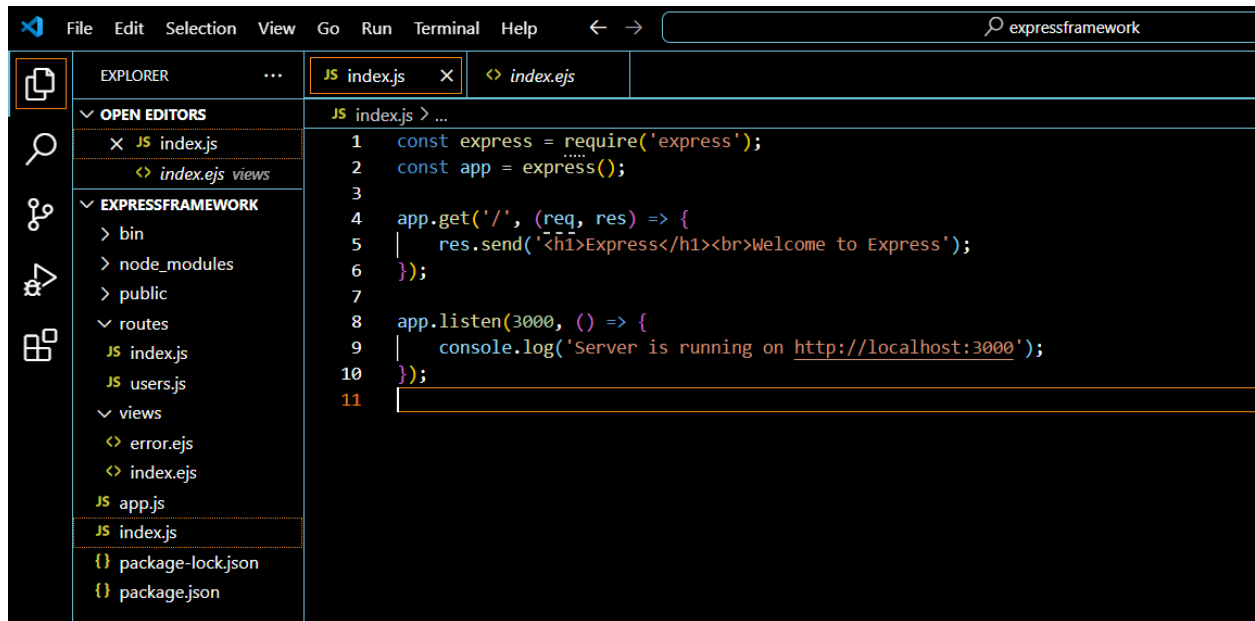


The screenshot shows the Visual Studio Code interface with the 'expressframework' directory open. The Explorer sidebar on the left displays the file structure, with 'index.ejs' selected under the 'views' folder. The main editor window shows the content of 'index.ejs', which is an HTML template. The terminal panel at the bottom shows the command 'npm install express ejs' and its output, including a security audit report.

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title><%= title %></title>
5     <link rel='stylesheet' href='/stylesheets/style.css
6   </head>
7   <body>
8     <h1><%= title %></h1>
9     <p>Welcome to <%= title %></p>
10  </body>
11 </html>
12
```

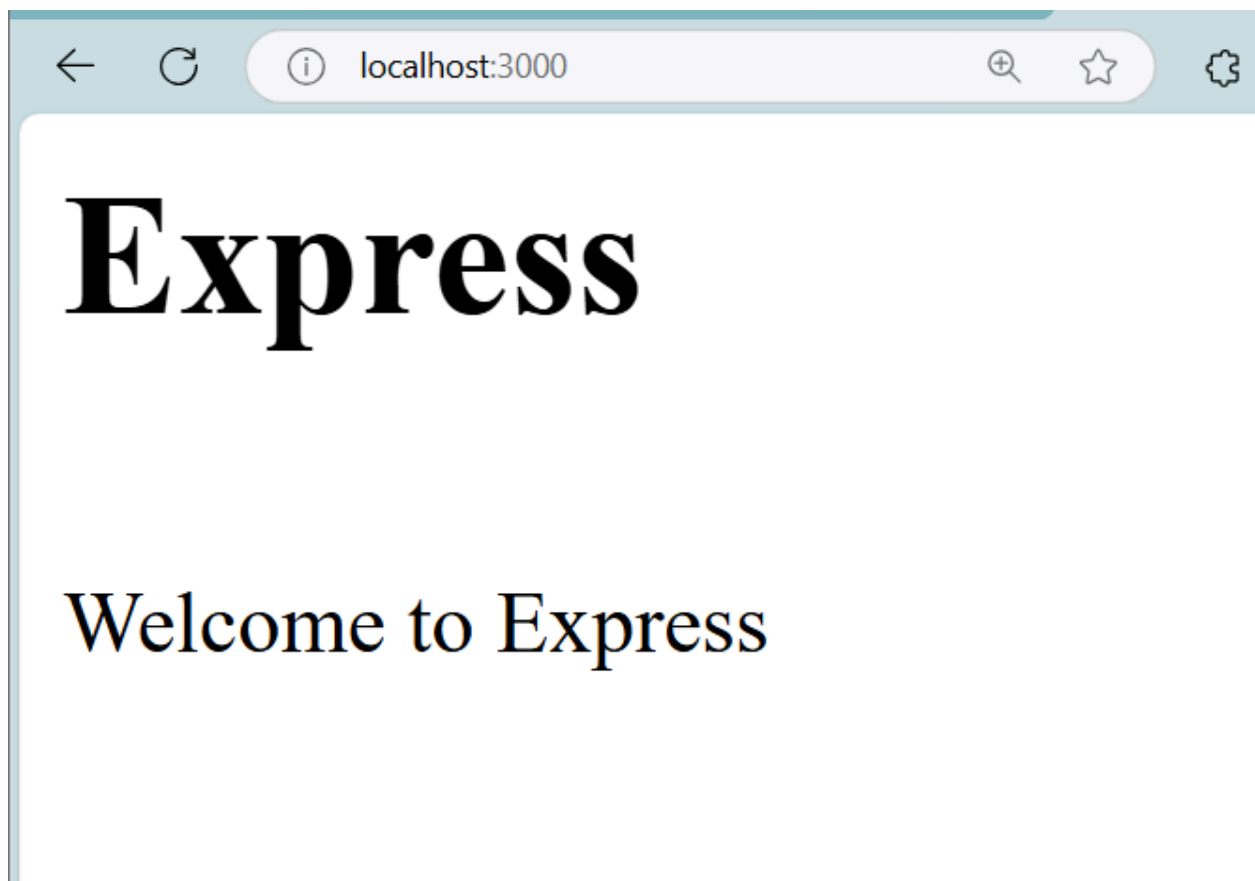
Terminal Output:

```
PS C:\Users\jessa\expressframework> npm install express ejs
added 54 packages, and audited 55 packages in 3m
8 vulnerabilities (3 low, 4 high, 1 critical)
To address all issues (including breaking changes), run:
  npm audit fix --force
Run `npm audit` for details.
PS C:\Users\jessa\expressframework>
```

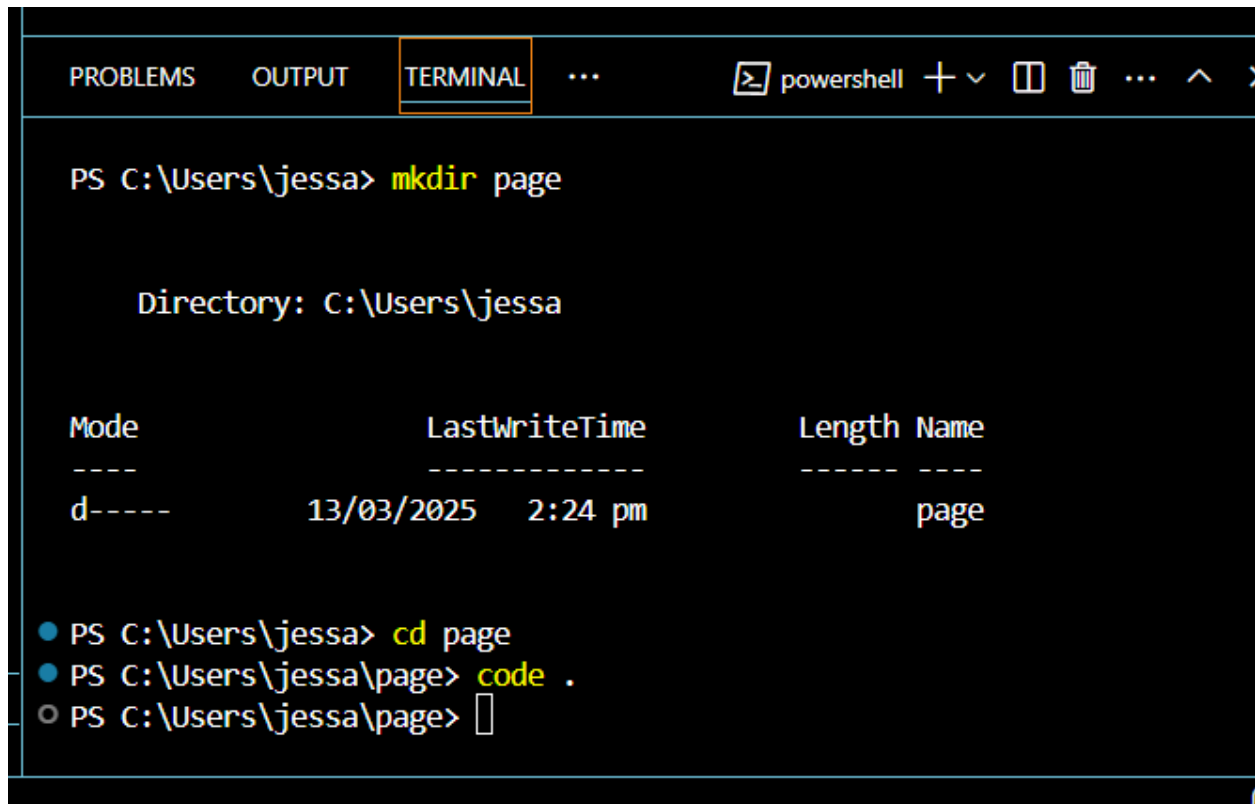


The image shows a Visual Studio Code editor window with the file explorer on the left and the code editor on the right. The file explorer shows the project structure, including the 'views' directory. The code editor displays the 'index.js' file, which contains the following code:

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => {
5   |   res.send('<h1>Express</h1><br>Welcome to Express');
6   | });
7
8 app.listen(3000, () => {
9   |   console.log('Server is running on http://localhost:3000');
10  | });
11
```



Task 3



The screenshot shows a PowerShell terminal window with the 'TERMINAL' tab selected. The prompt is 'PS C:\Users\jessa>'. The user enters 'mkdir page', and the output shows 'Directory: C:\Users\jessa' followed by a table of the new directory.

Mode	LastWriteTime	Length	Name
d-----	13/03/2025 2:24 pm		page

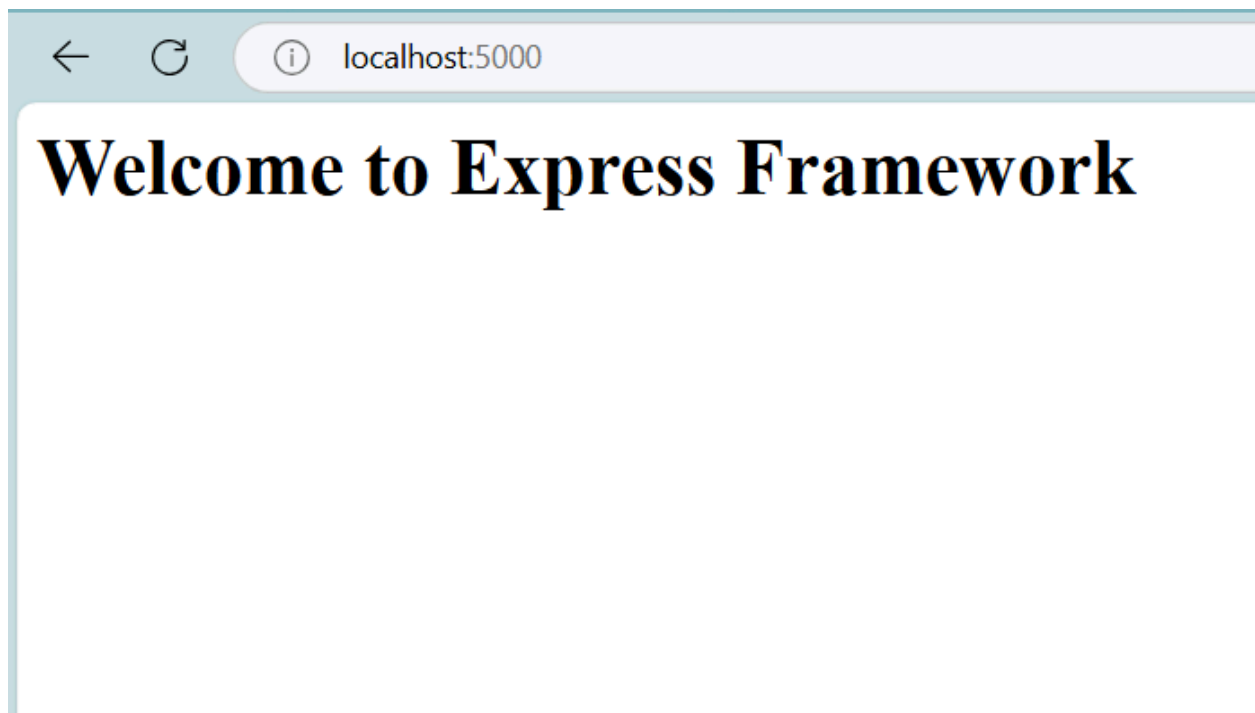
Below the table, the terminal shows the user navigating into the 'page' directory with 'cd page', opening the code editor with 'code .', and then pressing Enter.

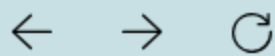
```
PS C:\Users\jessa> mkdir page

Directory: C:\Users\jessa

Mode                LastWriteTime         Length Name
----                -
d-----          13/03/2025   2:24 pm         page

• PS C:\Users\jessa> cd page
• PS C:\Users\jessa\page> code .
○ PS C:\Users\jessa\page> 
```





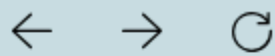
localhost:5000/about

About Us

Name: Jessa

School: Technological Institute of the Philippines

[Back to Home](#)



localhost:5000/contact

Contact Us

Email: jessa@example.com

Phone: 093-456-7890

[Back to Home](#)

5. Reflection

- **How straightforward was the process of setting up the Express.js application and defining routes for the Welcome, About Us, and Contact Us pages?**
 - Setting up the Express.js application and defining routes was fairly straightforward because the framework provides a structured way to manage routes. Initializing the project with `npm init` and installing dependencies was simple, and setting up routes in `app.js` was clear. The process became even easier after understanding how Express handles requests and responses.
- **Were there any challenges or roadblocks encountered during the implementation, such as errors or misconfigurations?**
 - Yes, I faced some challenges, especially with routing errors due to incorrect file paths and missing required modules. At first, I forgot to properly configure the `app.use()` function for static files, which caused issues in rendering views. Another problem was with EJS templates not displaying correctly due to syntax errors. However, reviewing the Express documentation and debugging the errors helped me fix these issues.
- **How effectively did the use of EJS templates facilitate the dynamic rendering of HTML content for the pages?**
 - Using EJS templates made it easier to manage dynamic content since it allowed me to pass data into the views without manually writing repetitive HTML. It helped in rendering different pages while keeping the code organized. Once I understood the syntax and how to use `<%= %>` for dynamic content, it became a powerful tool for managing my Express application.