

Express Web Server

Figure 1. Creating a new folder, creating a package, and installing express

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left showing a project structure for 'MIDTERMS' with folders 'express-rest' and 'node_modules', and files 'package-lock.json', 'package.json', and 'index.js'. The main editor area displays the 'index.js' file with the following code:

```
PS C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS> cd express-rest
PS C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS\express-rest> npm init --yes
wrote to C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS\express-rest\package.json:

{
  "name": "express-rest",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}

PS C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS\express-rest> npm i express

added 64 packages, and audited 65 packages in 6s

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

found 0 vulnerabilities
PS C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS\express-rest>
```

The terminal output shows the successful installation of Express.js. The status bar at the bottom indicates the file encoding is UTF-8 and the language is JavaScript.

Figure 2. Code for the Express Application

The screenshot shows the Visual Studio Code interface with the Explorer sidebar on the left showing a project structure for 'MIDTERMS' with folders 'express-rest' and 'node_modules', and files 'package-lock.json', 'package.json', and 'index.js'. The main editor area displays the 'index.js' file with the following code:

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => {
5   res.send('My New App!');
6 });
7
8 app.listen(3000, () => console.log('listening on port 3000'));
```

The terminal output shows the command to run the application:

```
PS C:\Users\ELIZA\Documents\EJ Files\EJ\College\3Y 1S\WCSEVER\MIDTERMS\express-rest> node index.js
listening on port 3000
```

The browser window on the left shows the output of the application: 'My New App!'. The status bar at the bottom indicates the file encoding is UTF-8 and the language is JavaScript.

Simulating a Simple API

Figure 3. Code for the resource

The screenshot shows a web browser at `localhost:3000/api/heroes` displaying a JSON array of superhero names: `["Superman", "Iron Man", "Batman", "Hulk"]`. In the background, VS Code is open with a project named `MIDTERMS`. The `index.js` file contains the following code:

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', (req, res) => {
5   res.send('My New App!');
6 });
7
8 app.get('/api/heroes', (req, res) => {
9   res.send(['Superman', 'Iron Man', 'Batman', 'Hulk']);
10 });
11
12
13 app.listen(3000, () => console.log('Listening on port 3000'));
```

The terminal shows the command `node index.js` being executed, resulting in the message: `Listening on port 3000`.

Install and use Nodemon Express

The screenshot shows VS Code with the same `index.js` file. The terminal now shows the command `nodemon index.js` being executed. The output indicates that Nodemon is installed and running the server, with the message: `Listening on port 3000`.

Express Route Parameters

Figure 4. Single params code

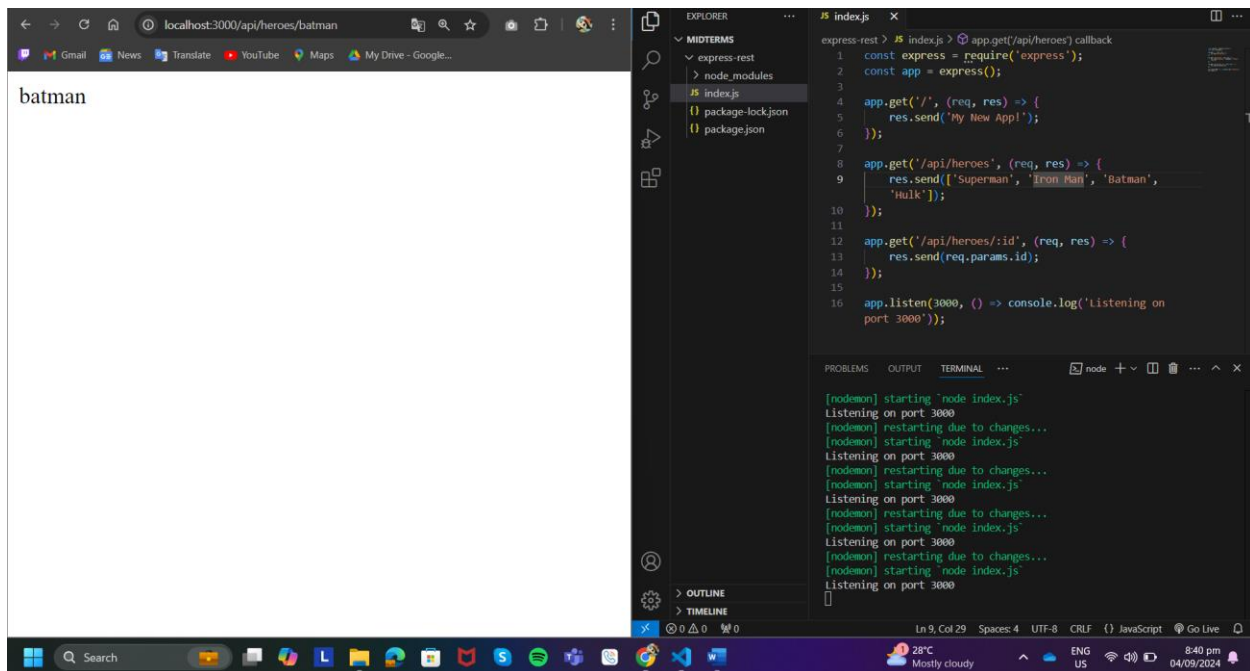


Figure 5. Multi params code

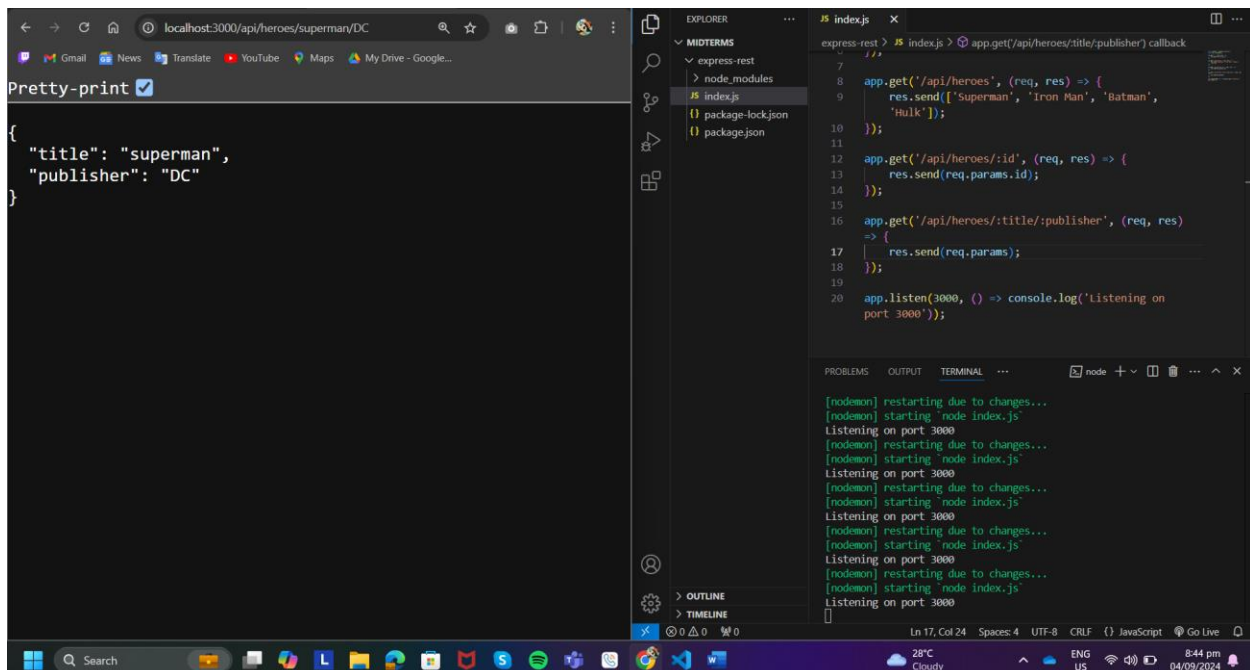
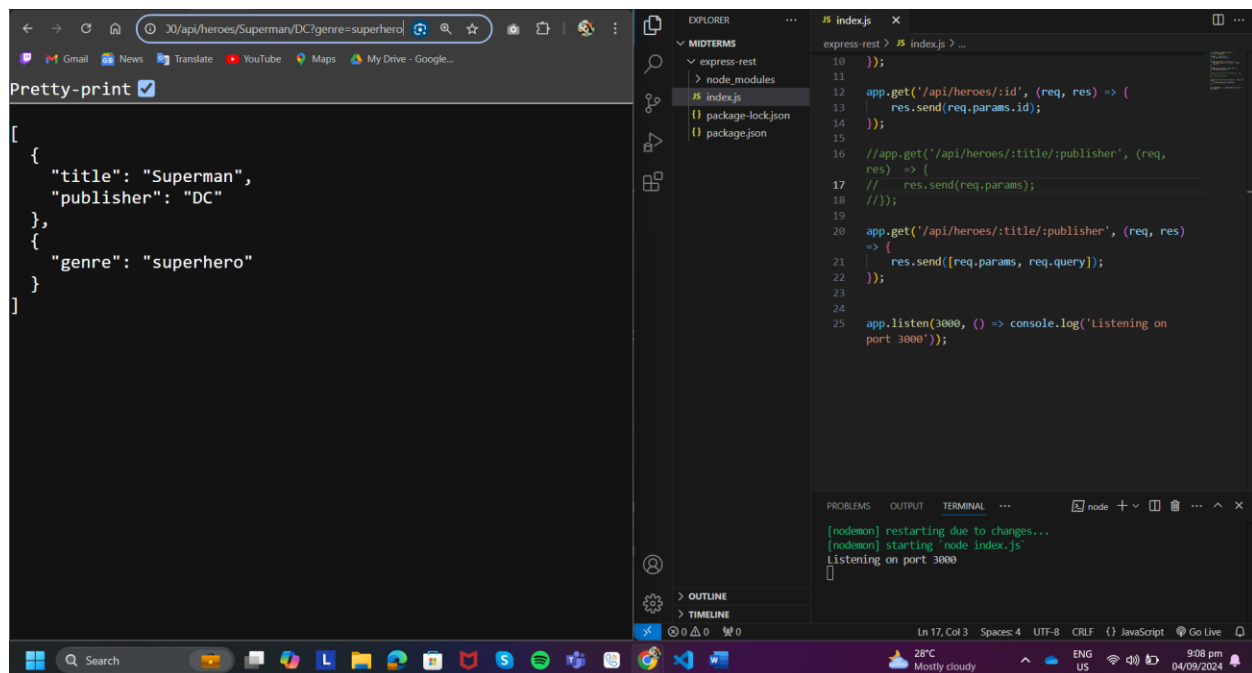


Figure 6. Query parameter code



The screenshot shows a web browser at the URL `30/api/heroes/Superman/DC?genre=superhero`. The response is a JSON array containing an object with the following properties:

```
[
  {
    "title": "Superman",
    "publisher": "DC"
  },
  {
    "genre": "superhero"
  }
]
```

Below the browser, the VS Code editor shows the `index.js` file with the following code:

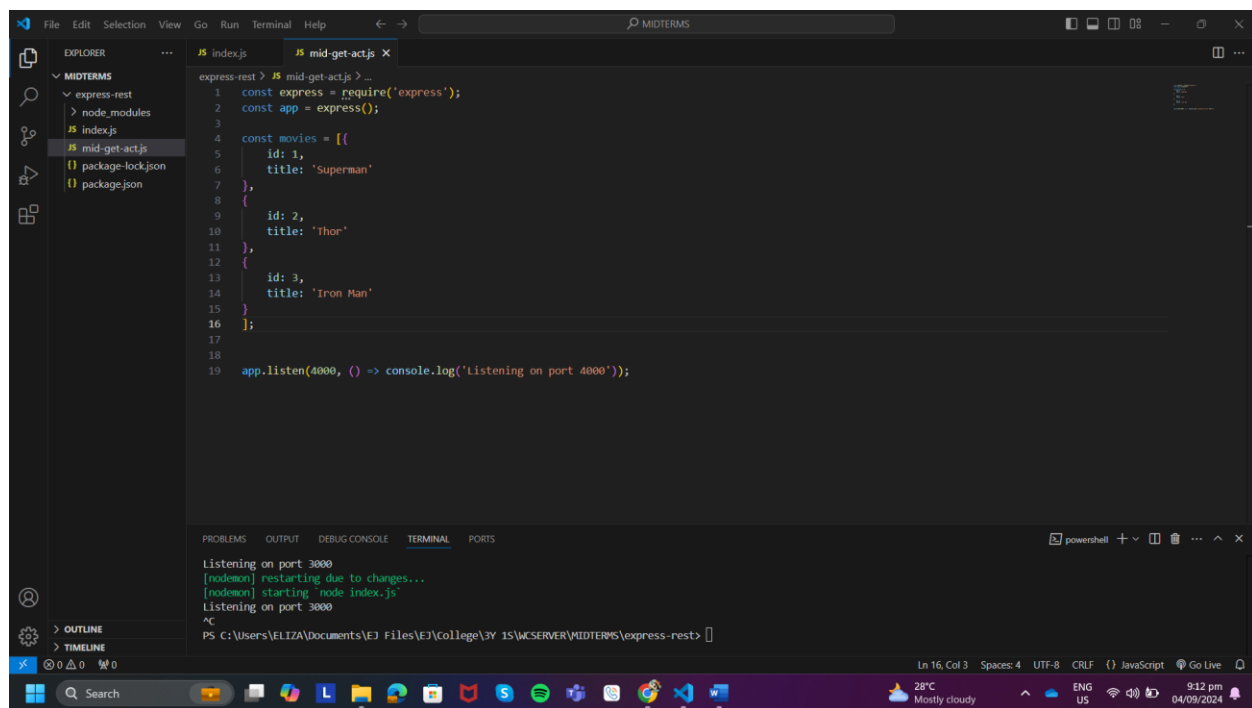
```
10  });
11
12  app.get('/api/heroes/:id', (req, res) => {
13    | res.send(req.params.id);
14  });
15
16  //app.get('/api/heroes/:title/:publisher', (req,
17  | res) => {
18  | // res.send(req.params);
19  | //});
20
21  app.get('/api/heroes/:title/:publisher', (req, res)
22  => {
23    | res.send([req.params, req.query]);
24  });
25
26  app.listen(3000, () => console.log('Listening on
port 3000'));
```

The terminal at the bottom shows the following output:

```
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Listening on port 3000
```

HTTP GET Request

Figure 7. Array for the GET Method



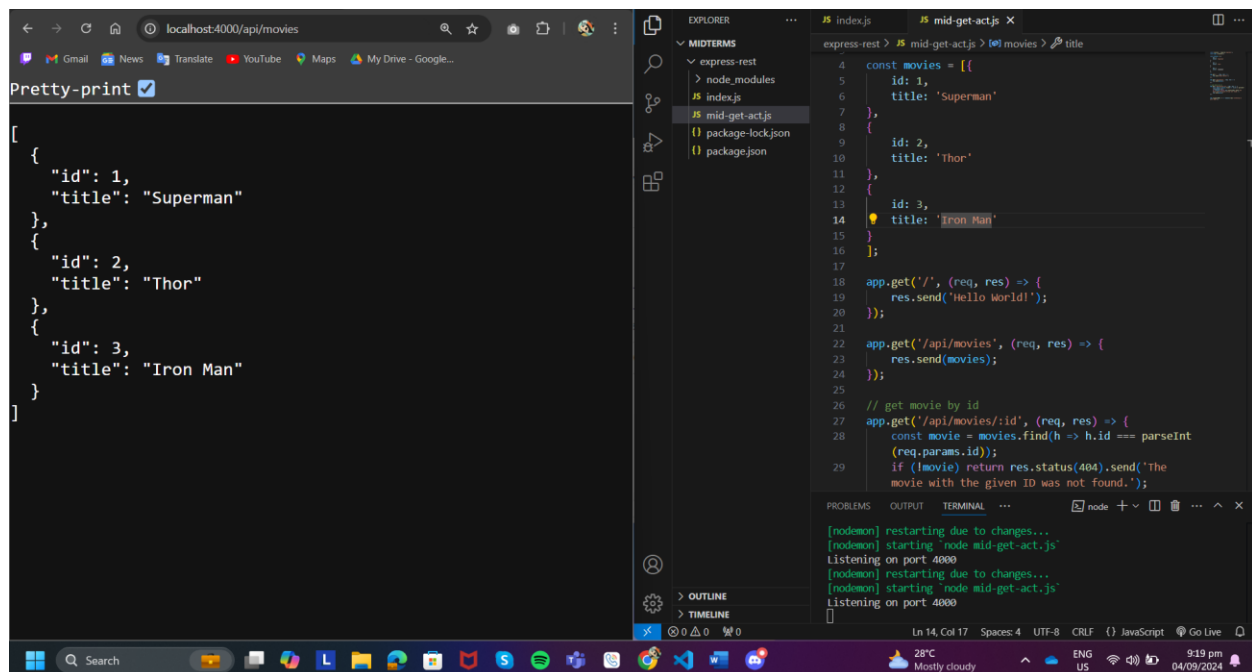
The screenshot shows the VS Code editor with the `mid-get-actjs` file open. The code defines a `movies` array and sets up an Express server listening on port 4000:

```
1  const express = require('express');
2  const app = express();
3
4  const movies = [
5    {
6      id: 1,
7      title: 'Superman'
8    },
9    {
10     id: 2,
11     title: 'Thor'
12   },
13   {
14     id: 3,
15     title: 'Iron Man'
16   }
17 ];
18
19 app.listen(4000, () => console.log('Listening on port 4000'));
```

The terminal at the bottom shows the following output:

```
Listening on port 3000
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Listening on port 3000
PS C:\Users\ELIZA\Documents\EJ Files\EJ\college\3Y 15\WCSEVER\MIDTERMS\express-rest>
```

Figure 8. Code to GET All Movies



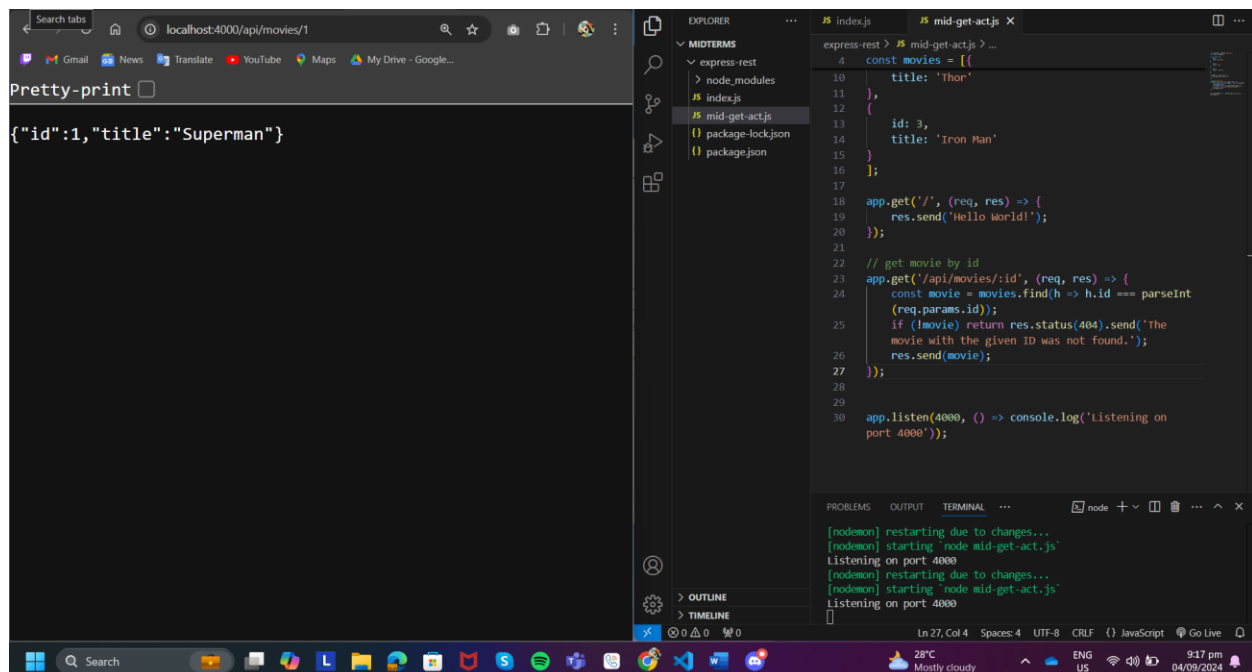
```
localhost:4000/api/movies
Pretty-print ☒
[
  {
    "id": 1,
    "title": "Superman"
  },
  {
    "id": 2,
    "title": "Thor"
  },
  {
    "id": 3,
    "title": "Iron Man"
  }
]

EXPLORER
mid-terms
  express-rest
    > node_modules
  index.js
  mid-get-act.js
  package-lock.json
  package.json

index.js
express-rest > mid-get-act.js > movies > title
4 const movies = [
5   {
6     id: 1,
7     title: 'Superman'
8   },
9   {
10    id: 2,
11    title: 'Thor'
12  },
13  {
14    id: 3,
15    title: 'Iron Man'
16  }
17 ];
18 app.get('/', (req, res) => {
19   res.send('Hello World!');
20 });
21
22 app.get('/api/movies', (req, res) => {
23   res.send(movies);
24 });
25
26 // get movie by id
27 app.get('/api/movies/:id', (req, res) => {
28   const movie = movies.find(h => h.id === parseInt
29     (req.params.id));
30   if (!movie) return res.status(404).send('The
31     movie with the given ID was not found.');
```

[nodemon] restarting due to changes...
[nodemon] starting 'node mid-get-act.js'
Listening on port 4000
[nodemon] restarting due to changes...
[nodemon] starting 'node mid-get-act.js'
Listening on port 4000

Figure 9. Code to find specific movie



```
localhost:4000/api/movies/1
Pretty-print ☐
{"id":1,"title":"Superman"}

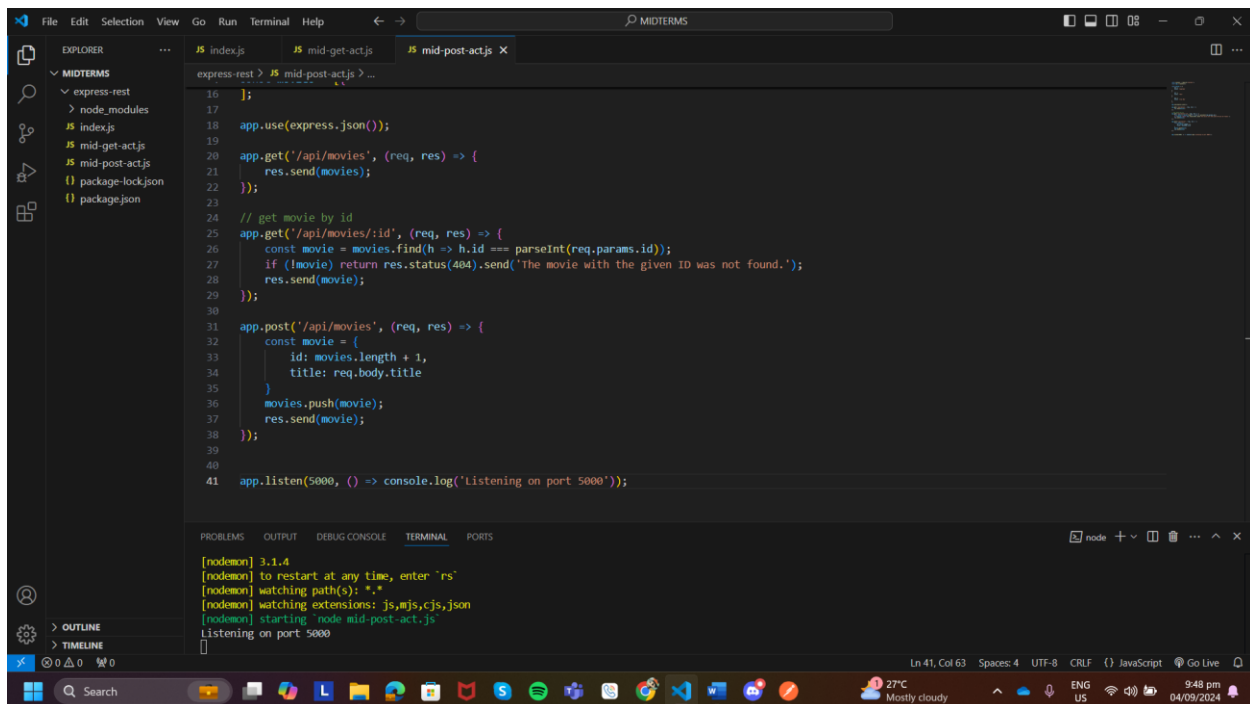
EXPLORER
mid-terms
  express-rest
    > node_modules
  index.js
  mid-get-act.js
  package-lock.json
  package.json

index.js
express-rest > mid-get-act.js > ...
10 const movies = [
11   {
12     title: 'Thor'
13   },
14   {
15     id: 3,
16     title: 'Iron Man'
17   }
18 ];
19
20 app.get('/', (req, res) => {
21   res.send('Hello World!');
22 });
23
24 // get movie by id
25 app.get('/api/movies/:id', (req, res) => {
26   const movie = movies.find(h => h.id === parseInt
27     (req.params.id));
28   if (!movie) return res.status(404).send('The
29     movie with the given ID was not found.');
```

[nodemon] restarting due to changes...
[nodemon] starting 'node mid-get-act.js'
Listening on port 4000
[nodemon] restarting due to changes...
[nodemon] starting 'node mid-get-act.js'
Listening on port 4000

HTTP POST Requests

Figure 10. POST Method to add element.



```
16  };
17
18  app.use(express.json());
19
20  app.get('/api/movies', (req, res) => {
21    res.send(movies);
22  });
23
24  // get movie by id
25  app.get('/api/movies/:id', (req, res) => {
26    const movie = movies.find(h => h.id === parseInt(req.params.id));
27    if (!movie) return res.status(404).send('The movie with the given ID was not found. ');
28    res.send(movie);
29  });
30
31  app.post('/api/movies', (req, res) => {
32    const movie = {
33      id: movies.length + 1,
34      title: req.body.title
35    };
36    movies.push(movie);
37    res.send(movie);
38  });
39
40
41  app.listen(5000, () => console.log('Listening on port 5000'));
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
[nodeemon] 3.1.4
[nodeemon] to restart at any time, enter "rs"
[nodeemon] watching path(s): *.*
[nodeemon] watching extensions: js,mjs,cjs,json
[nodeemon] starting "node mid-post-act.js"
Listening on port 5000
```

