

## Module 2: File System Module And Express.js

---

### Demo Document 1

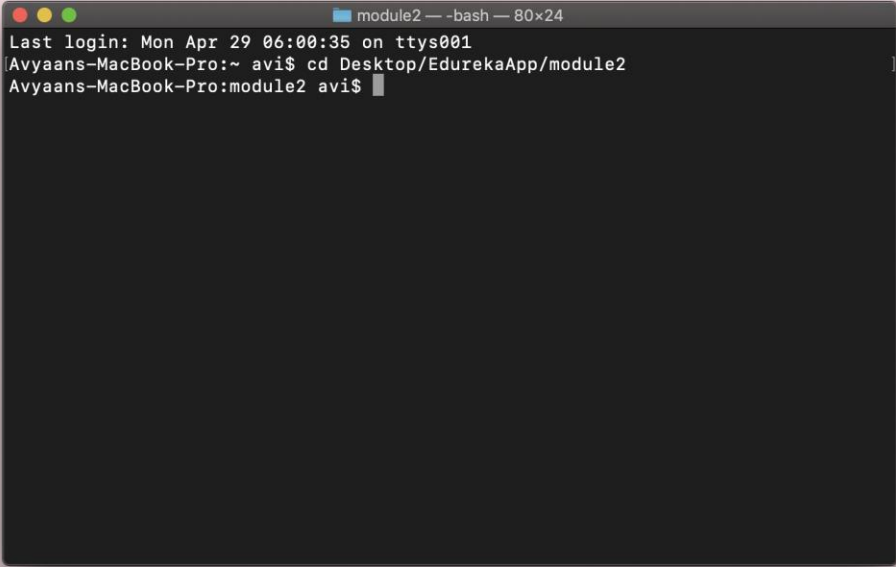
edureka!

**edureka!**

© Brain4ce Education Solutions Pvt. Ltd.

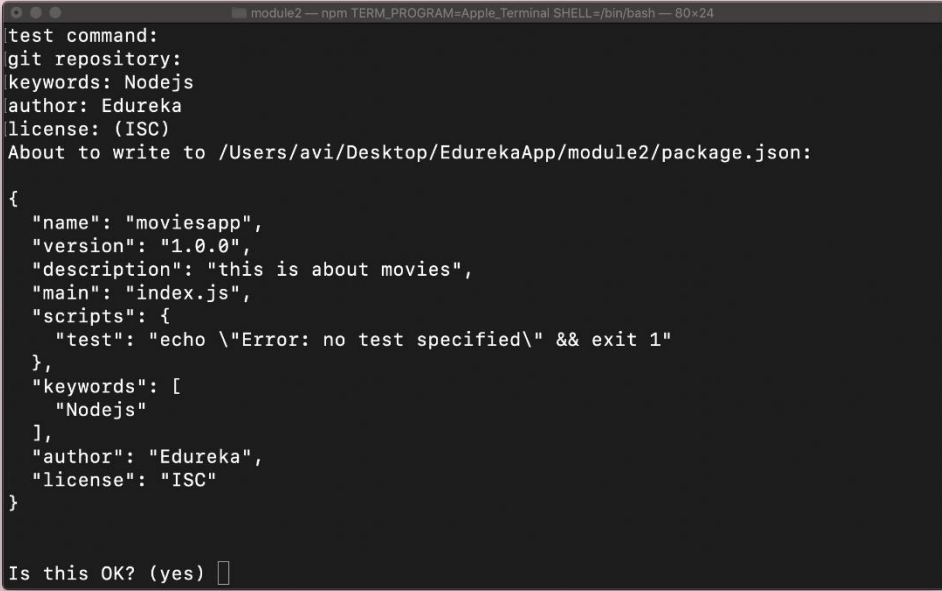
## Build an API using express, read a file with the FS module, and deploy the application using PM2 and Nginx

**Step 1:** Create a new folder in the system and navigate with the terminal or command prompt.



```
module2 — -bash — 80x24
Last login: Mon Apr 29 06:00:35 on ttys001
Avyaans-MacBook-Pro:~ avi$ cd Desktop/EdurekaApp/module2
Avyaans-MacBook-Pro:module2 avi$
```

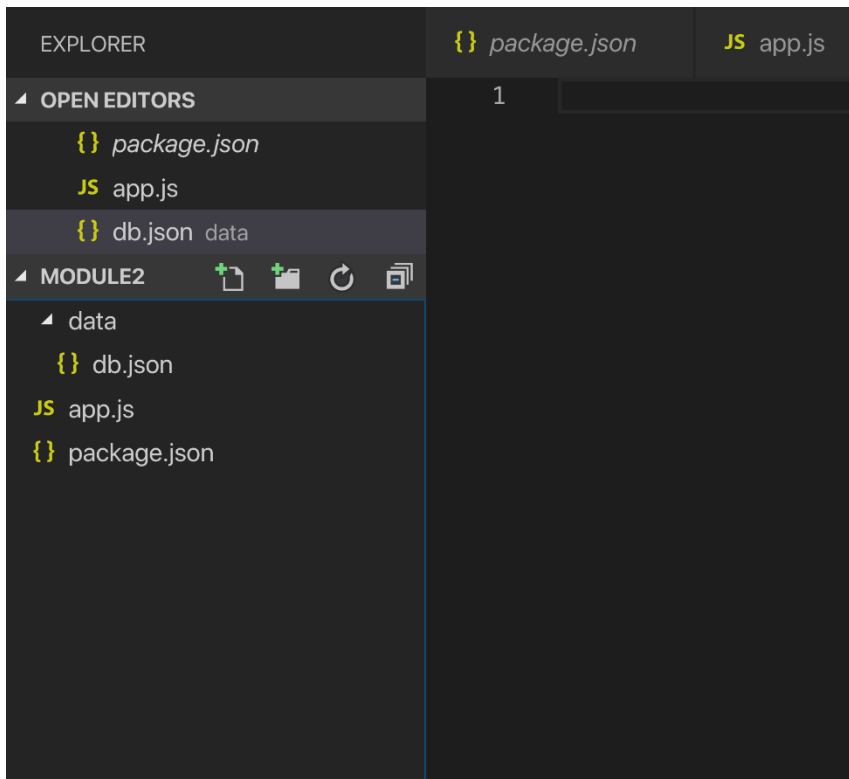
**Step 2:** Create ‘package.json’ inside the folder using command prompt with ‘npm init’ command and answer some questions regarding the app.



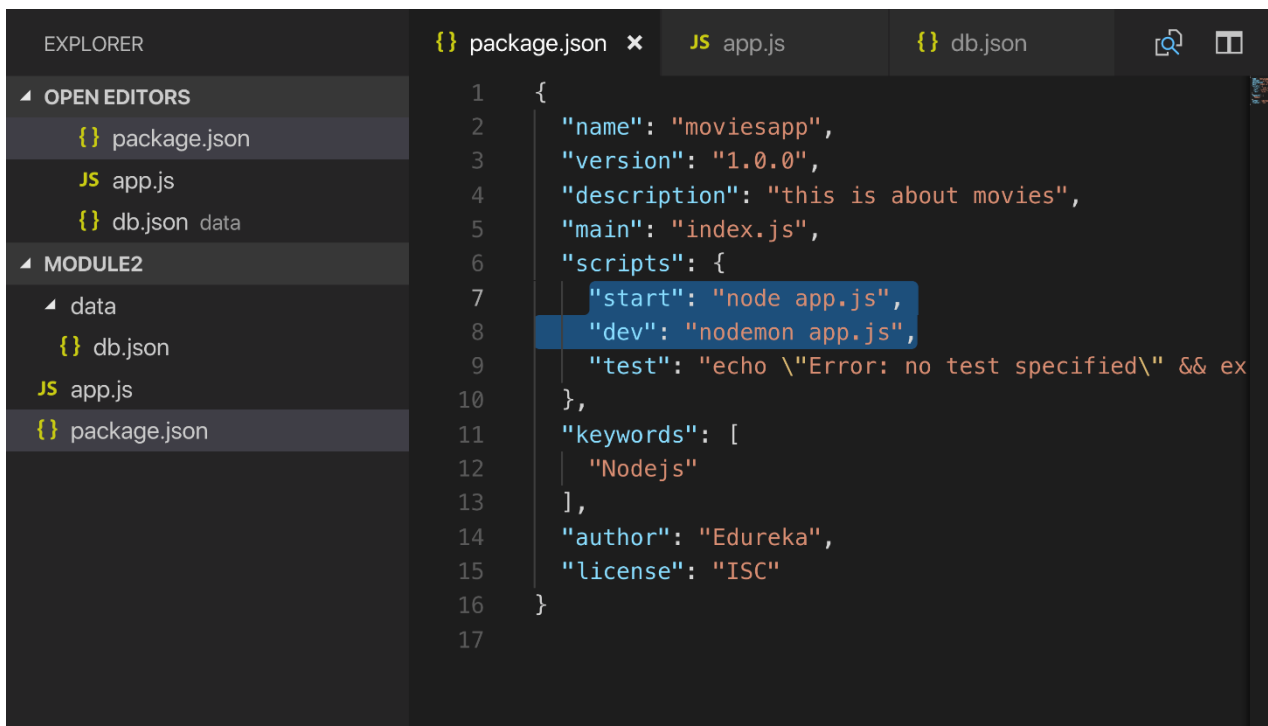
```
module2 — npm TERM_PROGRAM=Apple_Terminal SHELL=/bin/bash — 80x24
test command:
git repository:
keywords: Nodejs
author: Edureka
license: (ISC)
About to write to /Users/avi/Desktop/EdurekaApp/module2/package.json:
{
  "name": "moviesapp",
  "version": "1.0.0",
  "description": "this is about movies",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "Nodejs"
  ],
  "author": "Edureka",
  "license": "ISC"
}
Is this OK? (yes)
```

In the end, type ‘yes’ and a file with the name of package.json will be created.

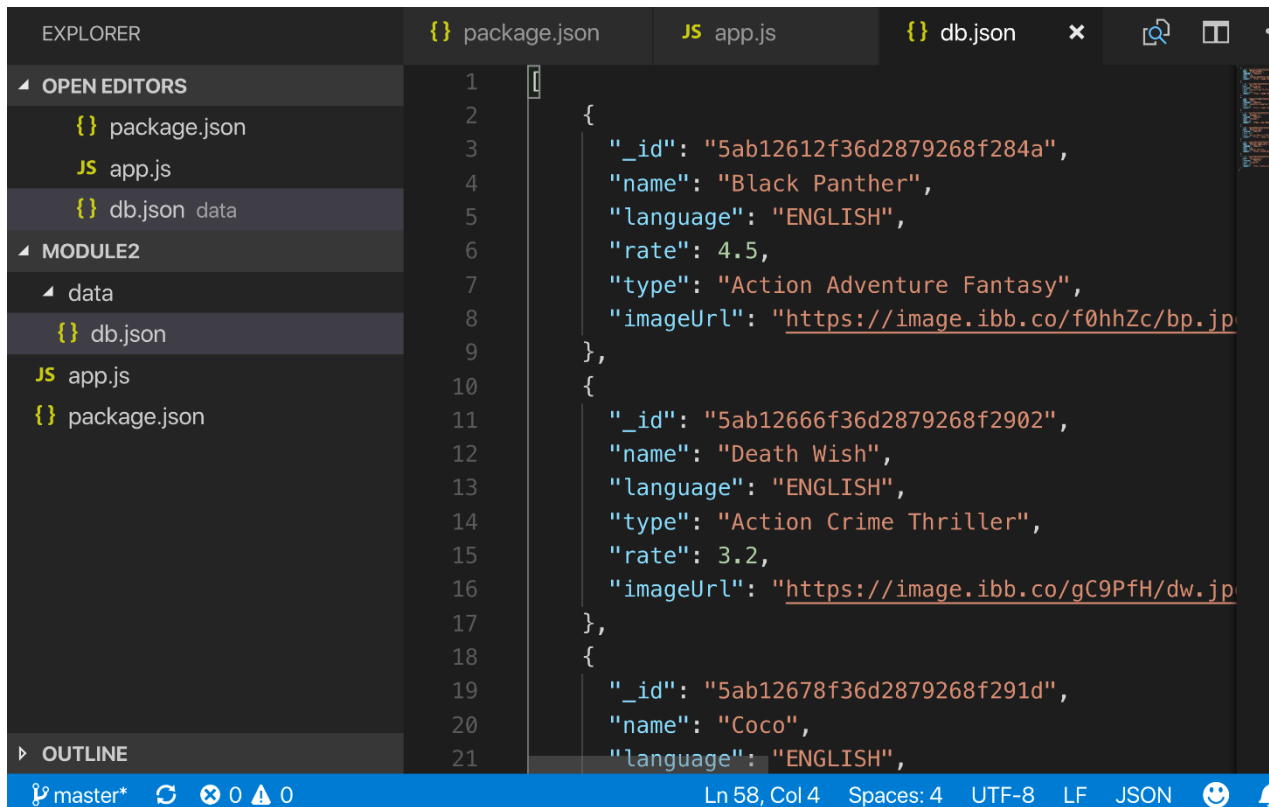
**Step 3:** Create a folder structure with one db folder to put the 'db.json' file with sample data and app.js for server code.



**Step 4:** Specify start and dev command in package.json to run the application on the production as well as the Development mode with node <filename>.



**Step 5:** Copy data into db.json using the db.json file present on LMS.



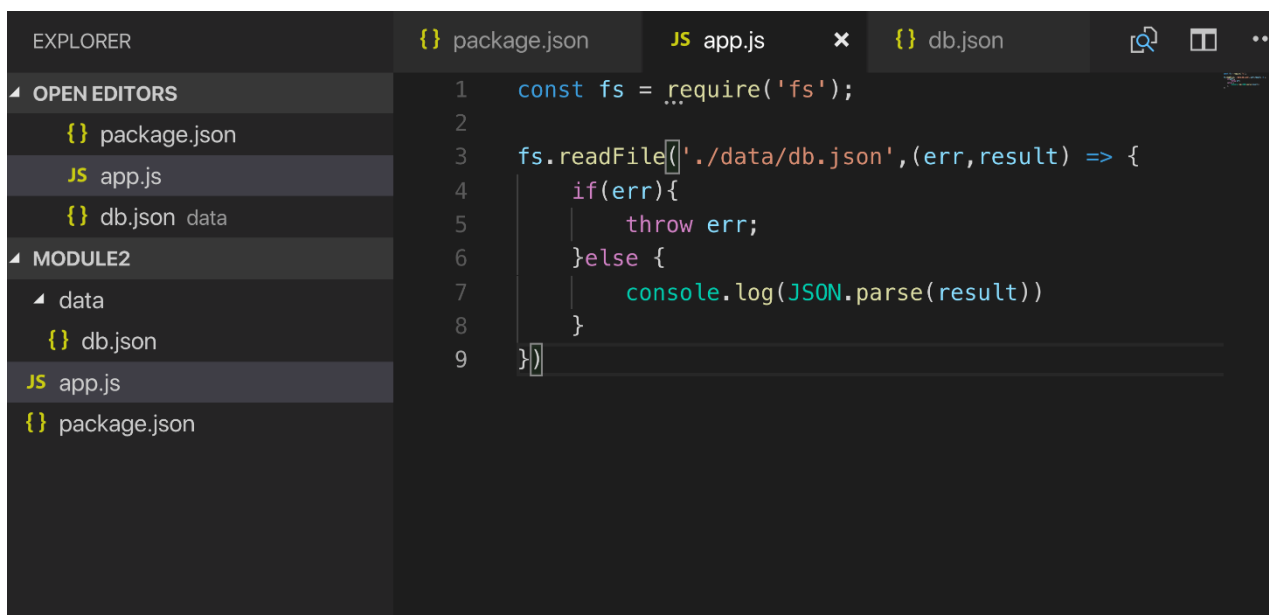
The screenshot shows the VS Code interface with the Explorer sidebar on the left. The Explorer shows a project structure with 'package.json', 'app.js', and 'db.json' in the root, and a 'data' folder containing 'db.json'. The main editor displays the content of 'db.json', which is a JSON array of three movie objects. The status bar at the bottom indicates 'Ln 58, Col 4'.

```

1  [
2    {
3      "_id": "5ab12612f36d2879268f284a",
4      "name": "Black Panther",
5      "language": "ENGLISH",
6      "rate": 4.5,
7      "type": "Action Adventure Fantasy",
8      "imageUrl": "https://image.ibb.co/f0hhZc/bp.jp
9    },
10   {
11     "_id": "5ab12666f36d2879268f2902",
12     "name": "Death Wish",
13     "language": "ENGLISH",
14     "type": "Action Crime Thriller",
15     "rate": 3.2,
16     "imageUrl": "https://image.ibb.co/gC9PfH/dw.jp
17   },
18   {
19     "_id": "5ab12678f36d2879268f291d",
20     "name": "Coco",
21     "language": "ENGLISH",

```

**Step 6:** Require Fs module in app.js and use *fs.readFile* to read data from db.json and console in the terminal window.



The screenshot shows the VS Code interface with the Explorer sidebar on the left. The Explorer shows a project structure with 'package.json', 'app.js', and 'db.json' in the root, and a 'data' folder containing 'db.json'. The main editor displays the content of 'app.js', which uses the 'fs' module to read the 'db.json' file. The status bar at the bottom indicates 'Ln 58, Col 4'.

```

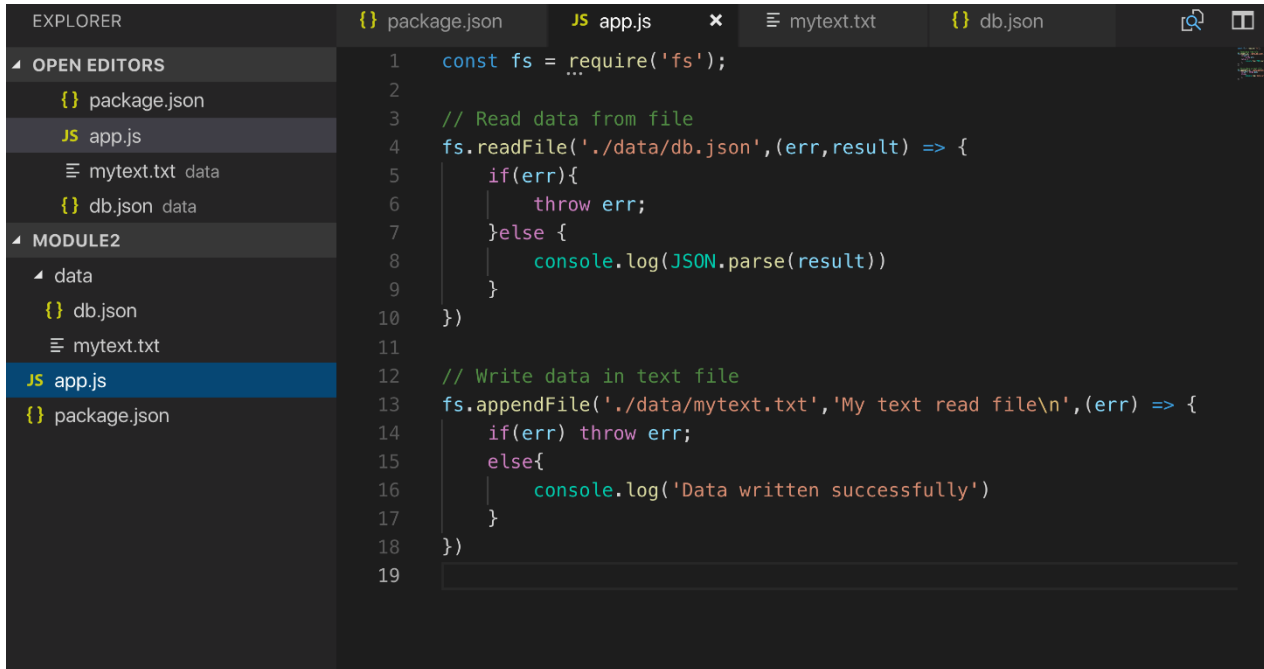
1  const fs = require('fs');
2
3  fs.readFile('./data/db.json', (err, result) => {
4    if(err){
5      throw err;
6    }else {
7      console.log(JSON.parse(result))
8    }
9  })

```

**Step 7:** Run "npm start" in command prompt to run the application and db.json will be sent out in the console window.

```
[nodemon] watching: *.*
[nodemon] starting 'node app.js'
[ { _id: '5ab12612f36d2879268f284a',
  name: 'Black Panther',
  language: 'ENGLISH',
  rate: 4.5,
  type: 'Action Adventure Fantasy',
  imageUrl: 'https://image.ibb.co/f0hhZc/bp.jpg' },
  { _id: '5ab12666f36d2879268f2902',
  name: 'Death Wish',
  language: 'ENGLISH',
  type: 'Action Crime Thriller',
  rate: 3.2,
  imageUrl: 'https://image.ibb.co/gC9PfH/dw.jpg' },
  { _id: '5ab12678f36d2879268f291d',
  name: 'Coco',
  language: 'ENGLISH',
  type: 'Adventure Animation Family',
  rate: 5,
  imageUrl: 'https://image.ibb.co/dQwWSx/coco.jpg' },
  { _id: '5ab126b6f36d2879268f2943',
  name: 'Avengers',
  language: 'ENGLISH',
  type: 'Action',
  rate: 2,
  imageUrl:
    'https://www.hindustantimes.com/rf/image_size_960x540/HT/p2/2018/04/01/Pictures/_46a0b2c0-3590-11e8-8c5f-3c6cc031651',
  },
  { _id: '5ab4e66b0c1d2b27846c6407',
  name: 'Black Friday',
  language: 'ENGLISH',
  rate: 4.5,
  type: 'Action Adventure Fantasy',
  imageUrl: 'https://image.ibb.co/f0hhZc/bp.jpg' },
  { _id: '5ab12686f36d2879268f2930',
  name: 'Mission Impossible',
  language: 'English',
  rate: 2.5,
  type: 'Horror Thriller',
  imageUrl:
    'https://pre00.deviantart.net/5d3b/th/pre/f/2017/313/2/b/mission_impossible__dark_directive_teaser_poster_by_themadb',
  },
  { _id: '5ab12698f36d2879268f293e',
  name: 'Incredibles 2',
  language: 'ENGLISH',
  type: 'Animated',
  rate: 4,
  imageUrl:
    'http://static1.squarespace.com/static/588a4776f5e23132a09d23b2/588a4e91be65945e50a36c0e/5b24084baa4a999c88a9f277/15' },
  ]
[nodemon] clean exit - waiting for changes before restart
```

**Step 8:** After read operation, we will use ***fs.appendFile*** to write in a text file in the data folder. ***fs.appendFile*** will append new data in the same file, in case of ***fs.writeFile***, it will override the data.




```

1  const fs = require('fs');
2
3  // Read data from file
4  fs.readFile('./data/db.json', (err, result) => {
5      if(err){
6          throw err;
7      }else {
8          console.log(JSON.parse(result))
9      }
10 })
11
12 // Write data in text file
13 fs.appendFile('./data/mytext.txt', 'My text read file\n', (err) => {
14     if(err) throw err;
15     else{
16         console.log('Data written successfully')
17     }
18 })
19

```

**Step 9:** Again, use the same command “npm start” or “npm run dev” we will see console data written successfully in the node console.

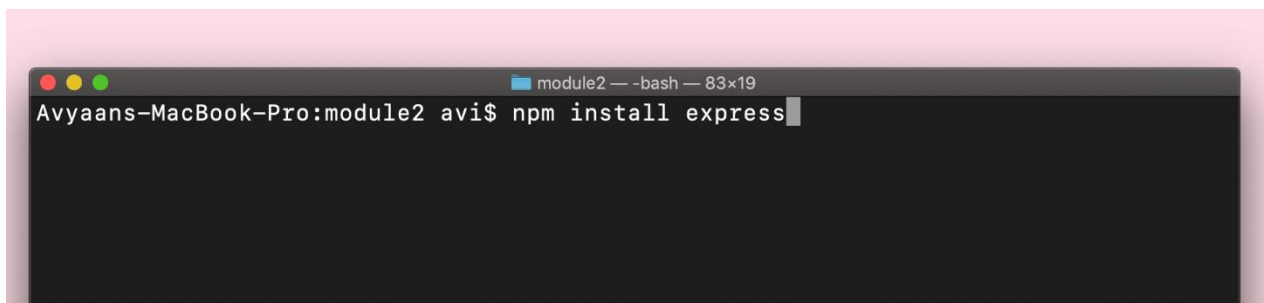


```

Data written successfully
[nodemon] clean exit - waiting for changes before restart

```

**Step 10:** To start creating a server with express, first of all install express in the same folder with the command “npm install express”.



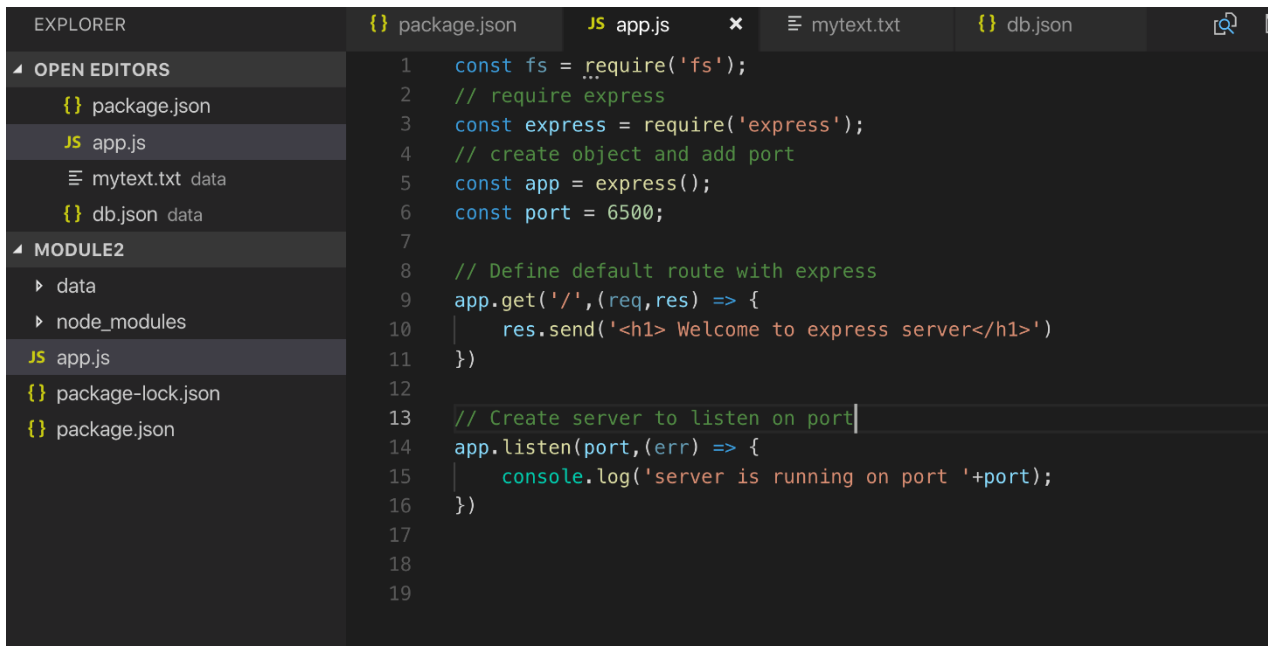
```

module2 -- -bash -- 83x19
Avyaans-MacBook-Pro:module2 avi$ npm install express

```

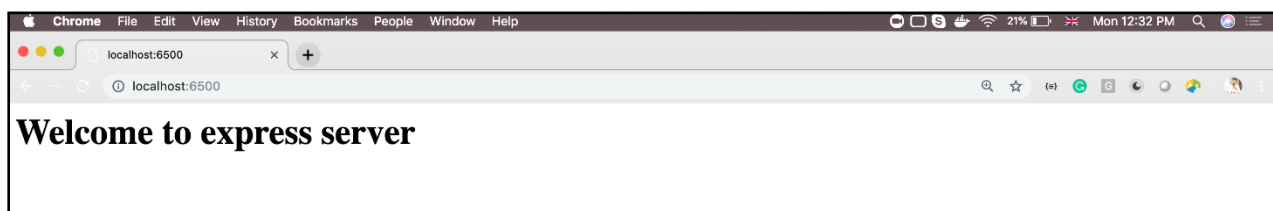
**Step 11:** Once installed, verify package must be added as dependency in package.json.

**Step 12:** Just like Fs require express also in app.js and create an object. Define the default route which will send the default message and make the server to listen on the port provided.

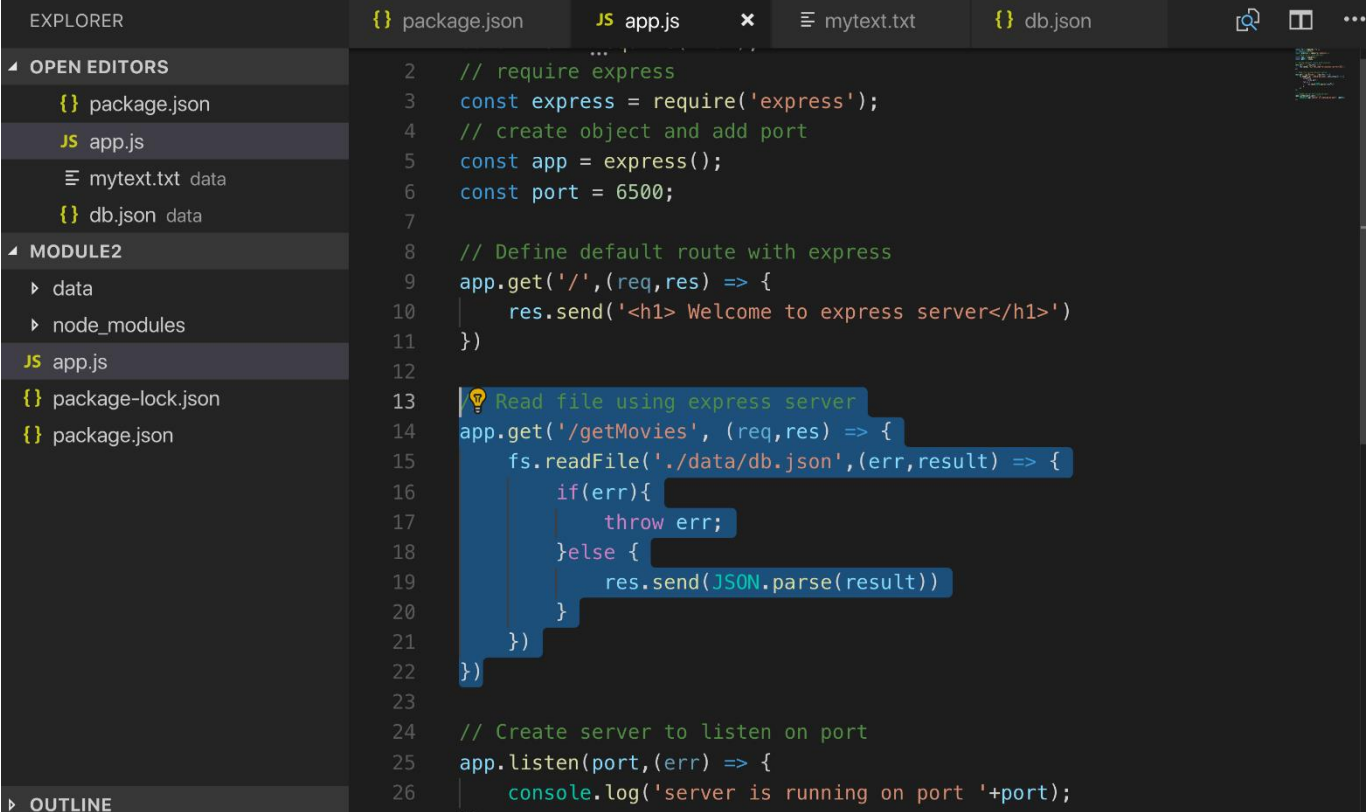


```
1  const fs = require('fs');
2  // require express
3  const express = require('express');
4  // create object and add port
5  const app = express();
6  const port = 6500;
7
8  // Define default route with express
9  app.get('/', (req, res) => {
10   |   res.send('<h1> Welcome to express server</h1>')
11   | })
12
13  // Create server to listen on port
14  app.listen(port, (err) => {
15   |   console.log('server is running on port '+port);
16   | })
17
18
19
```

**Step 13:** Again, run the application. To avoid start stop use “dev” command to start the application using “npm run dev” command. Open url localhost:6500 on the browser.



**Step 14:** Create one more route using express and add fs.readFile code in '/getMovies' route. This will help to read data from the file and send as json to the browser.



The screenshot shows the Visual Studio Code editor with the following components:

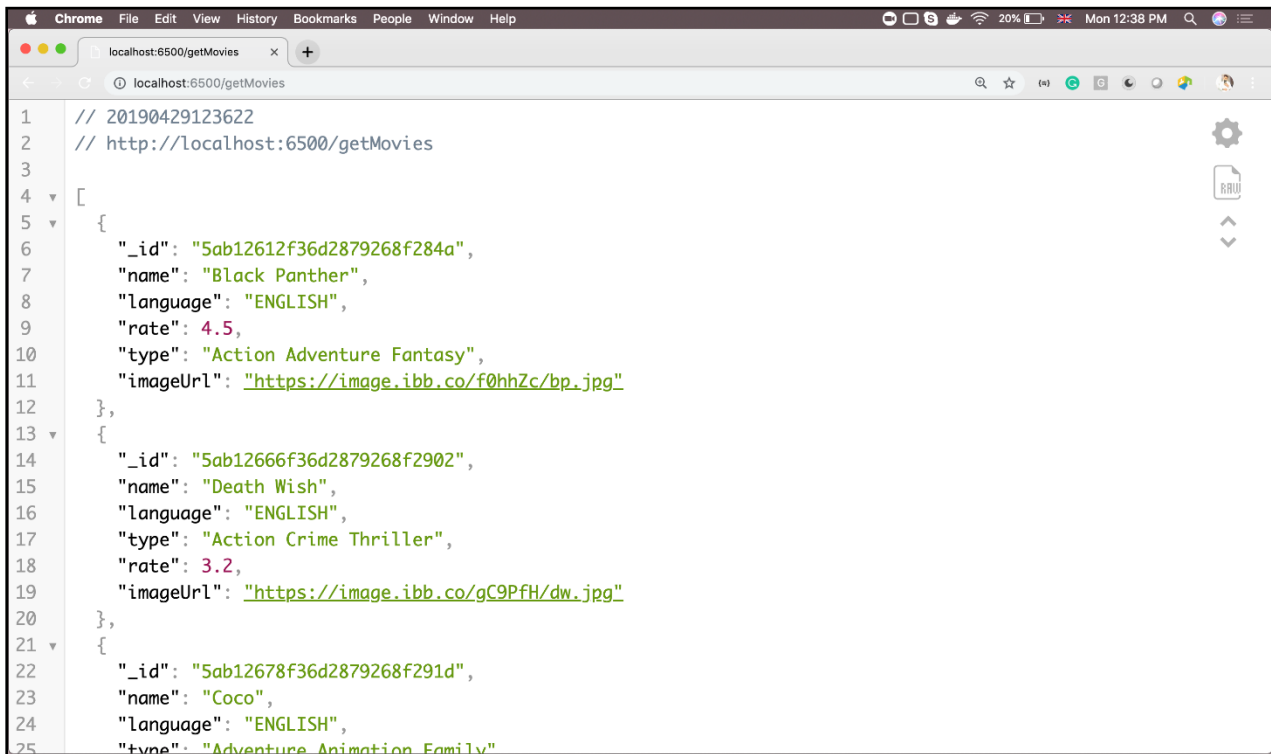
- EXPLORER:** Shows the project structure with 'MODULE2' containing 'data' and 'node\_modules'. The 'app.js' file is selected.
- APP.JS:** Contains the following code:

```
2 // require express
3 const express = require('express');
4 // create object and add port
5 const app = express();
6 const port = 6500;
7
8 // Define default route with express
9 app.get('/', (req, res) => {
10   res.send('<h1> Welcome to express server</h1>')
11 })
12
13 // Read file using express server
14 app.get('/getMovies', (req, res) => {
15   fs.readFile('./data/db.json', (err, result) => {
16     if(err){
17       throw err;
18     }else {
19       res.send(JSON.parse(result))
20     }
21   })
22 })
23
24 // Create server to listen on port
25 app.listen(port, (err) => {
26   console.log('server is running on port '+port);
```

In case of error, it will throw an error as well



**Step 15:** Finally, run “<http://localhost:6500/getMovies>” to see response on the browser. It will send JSON data saved in the file to the browser.



```
1 // 20190429123622
2 // http://localhost:6500/getMovies
3
4 [
5   {
6     "_id": "5ab12612f36d2879268f284a",
7     "name": "Black Panther",
8     "language": "ENGLISH",
9     "rate": 4.5,
10    "type": "Action Adventure Fantasy",
11    "imageUrl": "https://image.ibb.co/f0hhZc/bp.jpg"
12  },
13  {
14    "_id": "5ab12666f36d2879268f2902",
15    "name": "Death Wish",
16    "language": "ENGLISH",
17    "type": "Action Crime Thriller",
18    "rate": 3.2,
19    "imageUrl": "https://image.ibb.co/gC9PfH/dw.jpg"
20  },
21  {
22    "_id": "5ab12678f36d2879268f291d",
23    "name": "Coco",
24    "language": "ENGLISH",
25    "type": "Adventure Animation Family"
```

```
Avyaans-MacBook-Pro:code avi$ sudo npm install -g pm2
```

**Step 16:** Let’s run the app over Nginx and achieve reverse proxy  
Install “pm2”, this helps to run the application to run in the background and will keep it always running.

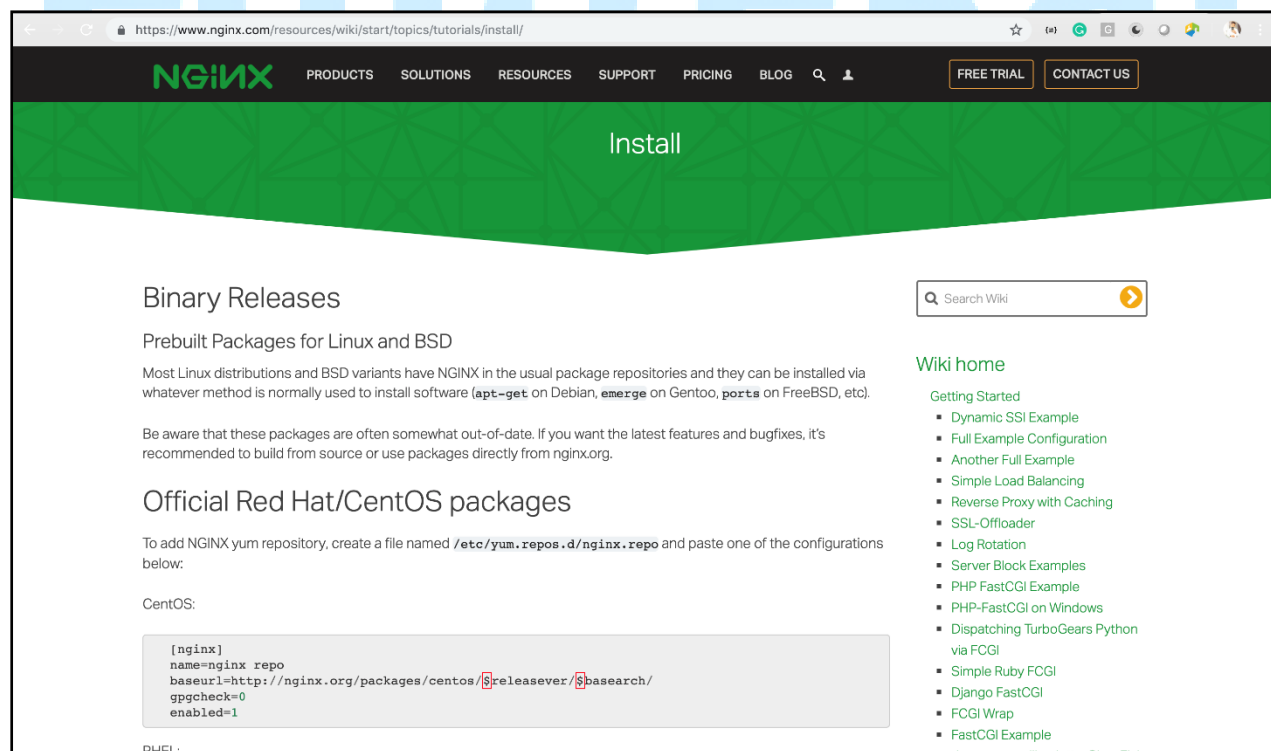
**Step 17:** Start the application using “pm2” and run in the background.  
To stop the application, use command “pm2 stop all”

```
Avyaans-MacBook-Pro:code avi$ pm2 start app.js
[PM2] Starting /Users/avi/Desktop/folder/EdurekaApp/module2/code/app.js in fork_mode (1 instance)
[PM2] Done.
```

App name	id	version	mode	pid	status	restart	uptime	cpu	mem	user	watching
app	1	1.0.0	fork	3903	online	0	0s	0%	7.0 MB	avi	disabled
start	0	1.0.0	fork	0	stopped	0	0	0%	0 B	avi	disabled

```
Use 'pm2 show <id|name>' to get more details about an app
Avyaans-MacBook-Pro:code avi$
```

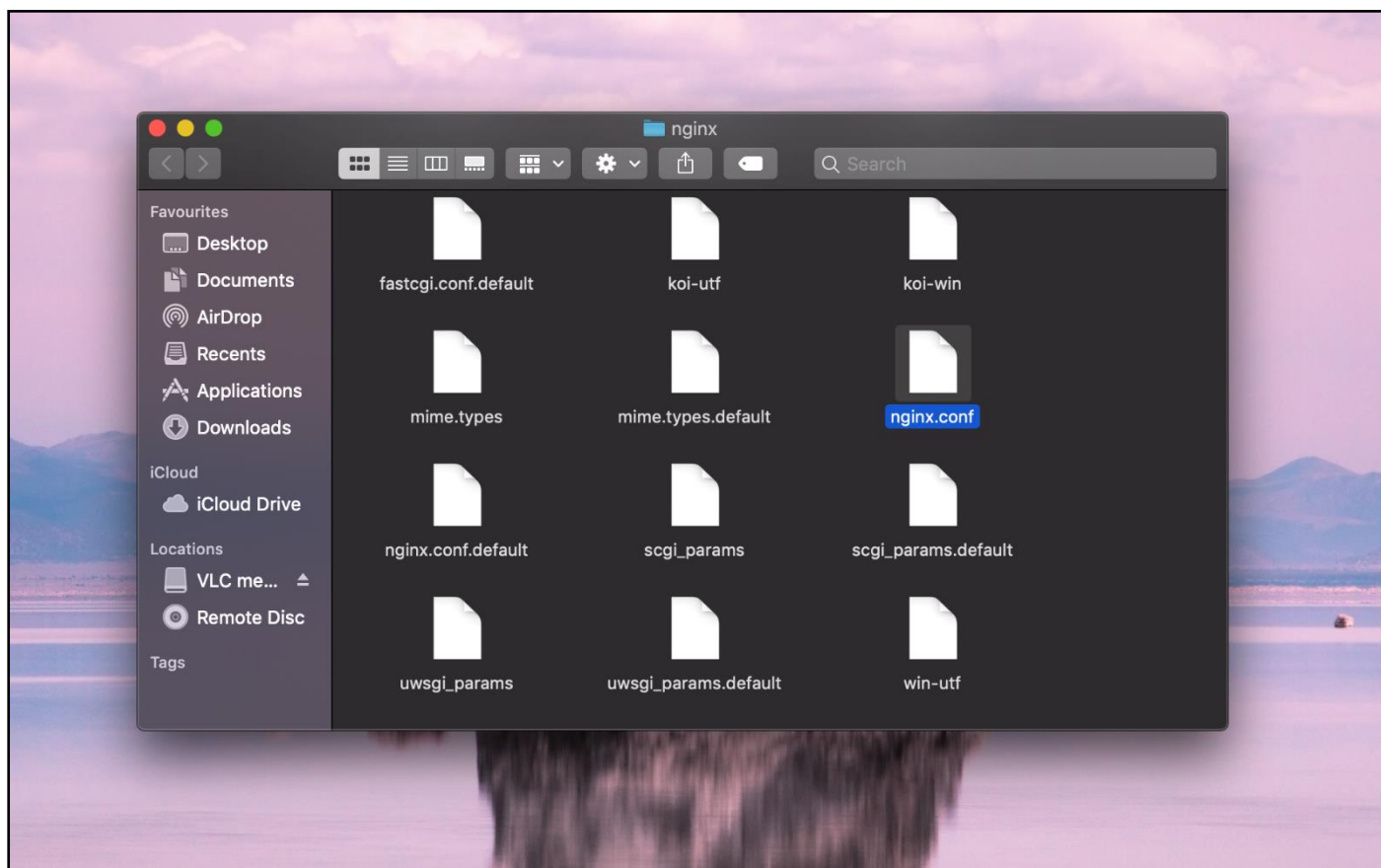
**Step 18:** Install Nginx for on the system  
Select the appropriate version according to OS.



**Step 19:** Navigate to the Nginx folder in your system and edit the nginx.conf file

Path for Mac: /usr/local/etc/nginx

Path for Windows: C:\nginx\conf



**Step 20:** Edit the Nginx.conf file and server key, add the path of running application over Pm2. And provide the path over which you want to run in reverse proxy.

```
#location ~ /\.php$ {
#    root           html;
#    fastcgi_pass   127.0.0.1:9000;
#    fastcgi_index  index.php;
#    fastcgi_param  SCRIPT_FILENAME  /scripts$fastcgi_script_name;
#    include        fastcgi_params;
#}

# deny access to .htaccess files, if Apache's document root
# concurs with nginx's one
#
#location ~ /\.ht {
#    deny all;
#}

# another virtual host using mix of IP-, name-, and port-based configuration
#
server {
    listen      8081;
    server_name localhost;

    location / {
        root   /var/www/react-app;
        index  index.html index.htm;
    }
}

server {
    listen      8082;
    server_name localhost;

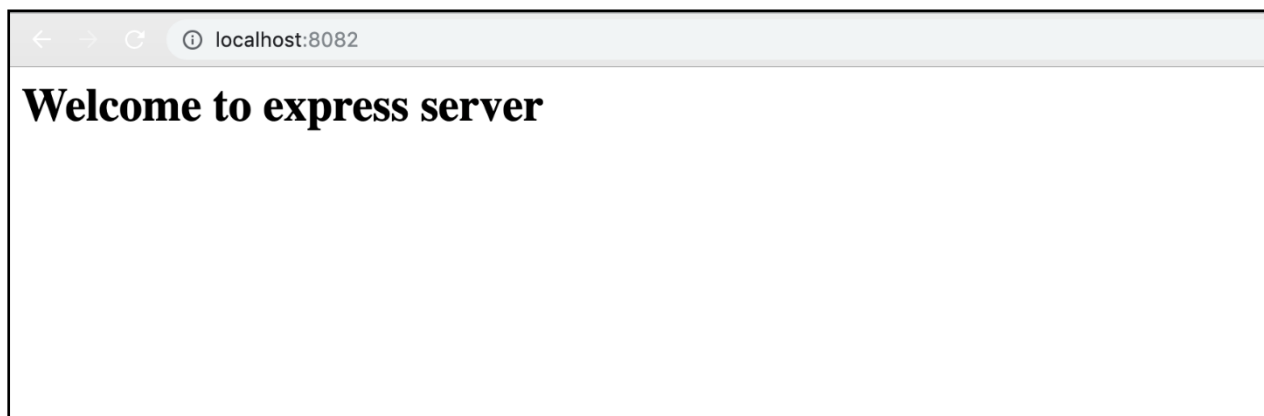
    location / {
        proxy_pass http://127.0.0.1:6500;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}

# HTTPS server
#
#server {
#    listen      443 ssl;
#    server_name localhost;

#    ssl_certificate      cert.pem;
#    ssl_certificate_key  cert.key;
```

Inside the Conf file, you can also set the SSL configuration.

**Step 21:** New port for running the application is localhost:8082 instead of localhost:6500 reverse proxy is implemented with Nginx



edureka!