

Module 9: Testing Node.js Application

Demo Document 1

edureka!

edureka!

© Brain4ce Education Solutions Pvt. Ltd.

Testing with Mocha and Chai

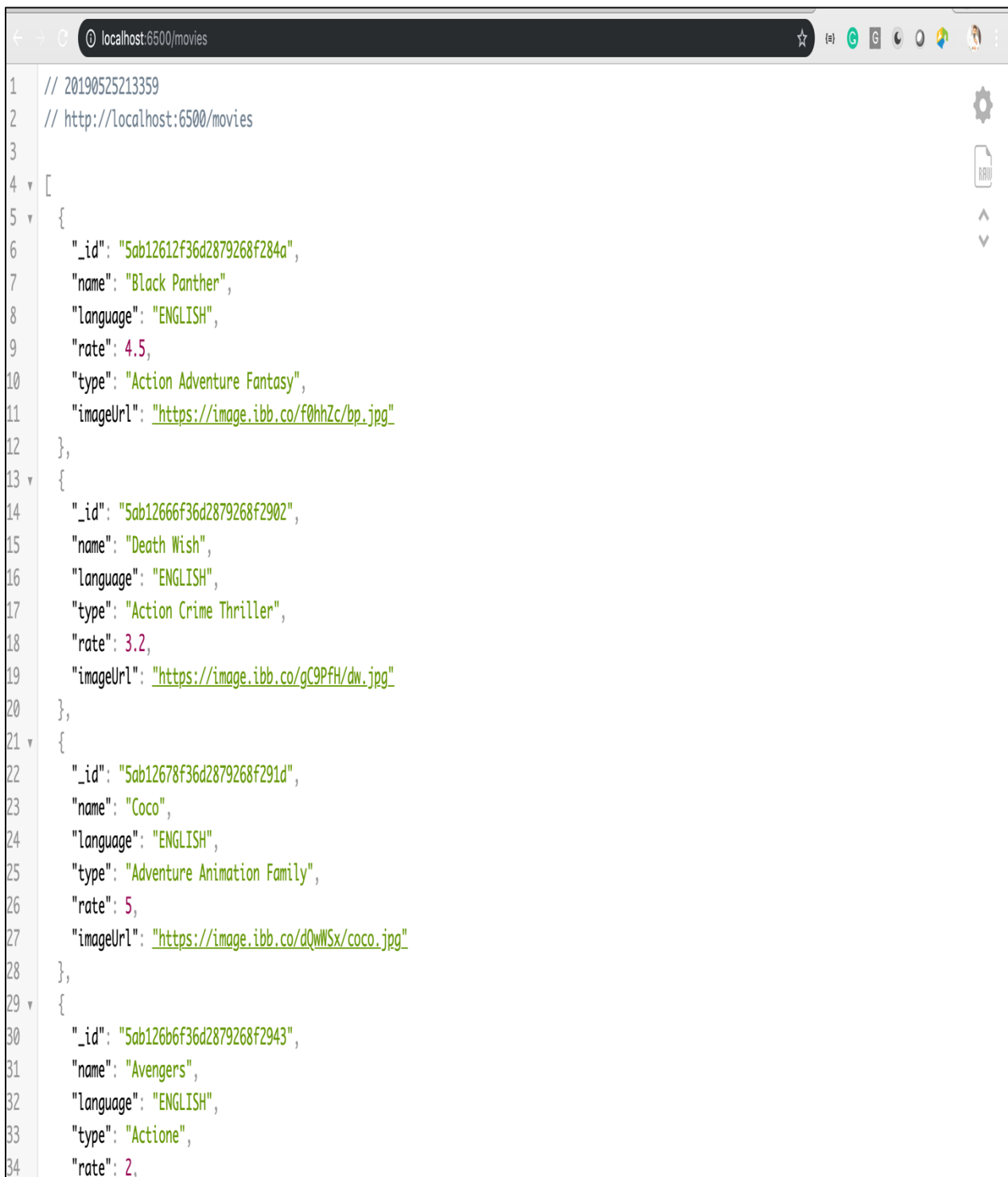
DEMO Steps:

Step 1: For testing the APIs with end points, create the APIs with all the CRUD operation.

```
1 import fs from 'fs';
2 import express from 'express';
3 const app = express();
4 const port = 6500;
5
6
7 app.get('/', (req, res) => {
8   res.send('<h1>Welcome to api For Fs</h1>')
9 });
10
11 app.get('/movies', (req, res) => {
12   fs.readFile('db.json', (err, result) => {
13     if(err) throw err;
14     res.send(JSON.parse(result));
15   })
16 })
17
18 app.get('/mytext', (req, res) => {
19   fs.readFile('myText.txt', 'utf-8', (err, data) => {
20     if(err) throw err;
21     res.send(data)
22   })
23 })
24
25 app.get('/bothops', (req, res) => {
26   fs.appendFile('mytext2.txt', 'My text read file\n', (err) => {
27     if(err) throw err;
28     else{
29       fs.readFile('mytext2.txt', 'utf-8', (err, data) => {
30         if(err) throw err;
31         res.send(data)
32       })
33     }
34   })
35 })
36
37
38 app.listen(port, (err) => {
39   console.log('Server is running on port ' + port)
40 });
```

Step 2: After creating APIs run that in browser, if you give a GET request, make sure that API is responding with status 200 Ok and we must be getting the API response

Module 9: Testing Node.js Application



The screenshot shows a web browser window with the address bar displaying 'localhost:6500/movies'. The main content area shows a JSON array of four movie objects. The first object is for 'Black Panther' with a rate of 4.5. The second is for 'Death Wish' with a rate of 3.2. The third is for 'Coco' with a rate of 5. The fourth is for 'Avengers' with a rate of 2. Each object includes an _id, name, language, type, and imageUrl.

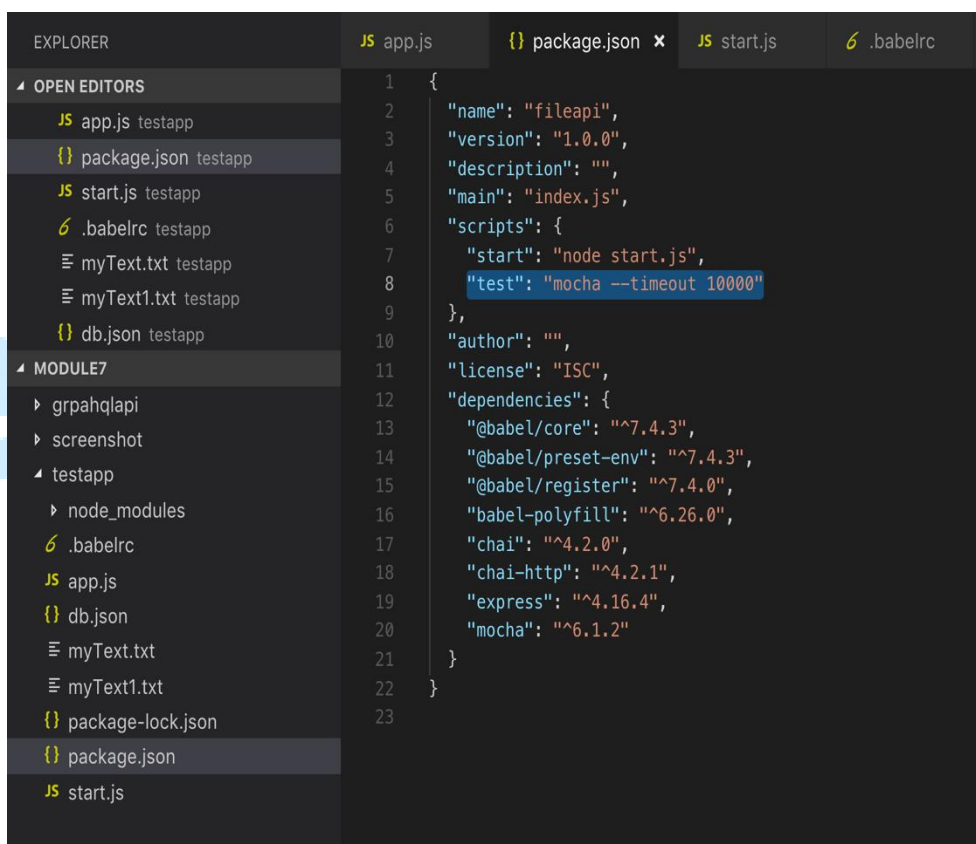
```
1 // 20190525213359
2 // http://localhost:6500/movies
3
4 [
5   {
6     "_id": "5ab12612f36d2879268f284a",
7     "name": "Black Panther",
8     "language": "ENGLISH",
9     "rate": 4.5,
10    "type": "Action Adventure Fantasy",
11    "imageUrl": "https://image.ibb.co/f0hhZc/bp.jpg"
12  },
13  {
14    "_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",
26    "rate": 5,
27    "imageUrl": "https://image.ibb.co/dQwWSx/coco.jpg"
28  },
29  {
30    "_id": "5ab126b6f36d2879268f2943",
31    "name": "Avengers",
32    "language": "ENGLISH",
33    "type": "Action",
34    "rate": 2,
```

Step 3: Install the below listed packages to test

“npm install mocha chai chai-http”

- ⇒ Api is created using express and @babel for es6
- ⇒ Now we need to add test command just like start
- ⇒ “test”:”mocha –timeout 1000”

As mocha act like a test runner and it will timeout after 1000sec without response.



The screenshot shows the VS Code editor interface. The Explorer panel on the left shows the project structure with files like app.js, package.json, start.js, and .babelrc. The main editor area displays the package.json file. The 'test' script is highlighted in blue, showing the command 'mocha --timeout 10000'. The dependencies section lists various packages including @babel/core, @babel/preset-env, @babel/register, babel-polyfill, chai, chai-http, express, and mocha.

```
1 {
2   "name": "fileapi",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "node start.js",
8     "test": "mocha --timeout 10000"
9   },
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "@babel/core": "^7.4.3",
14    "@babel/preset-env": "^7.4.3",
15    "@babel/register": "^7.4.0",
16    "babel-polyfill": "^6.26.0",
17    "chai": "^4.2.0",
18    "chai-http": "^4.2.1",
19    "express": "^4.16.4",
20    "mocha": "^6.1.2"
21  }
22 }
```

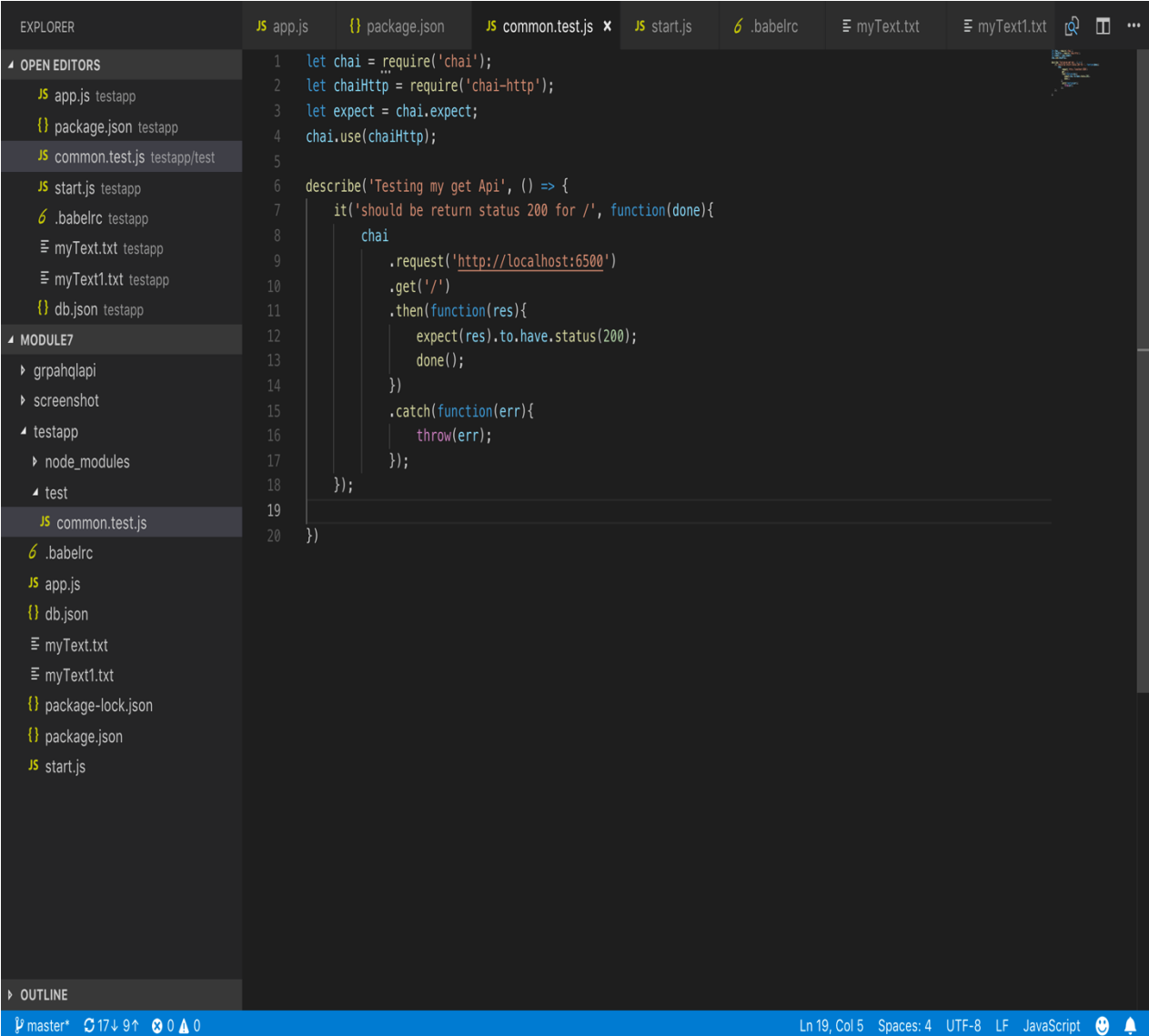
Step 4: To write a test case first we need to create a file with extension **.test.js**

Import chai and mocha,

In **describe**, add description of testcase and using assertion “**it**” we need to add callback function, using chai write test case with promise (.then), where we expect a response with status 200.

And with catch we can perform error handling

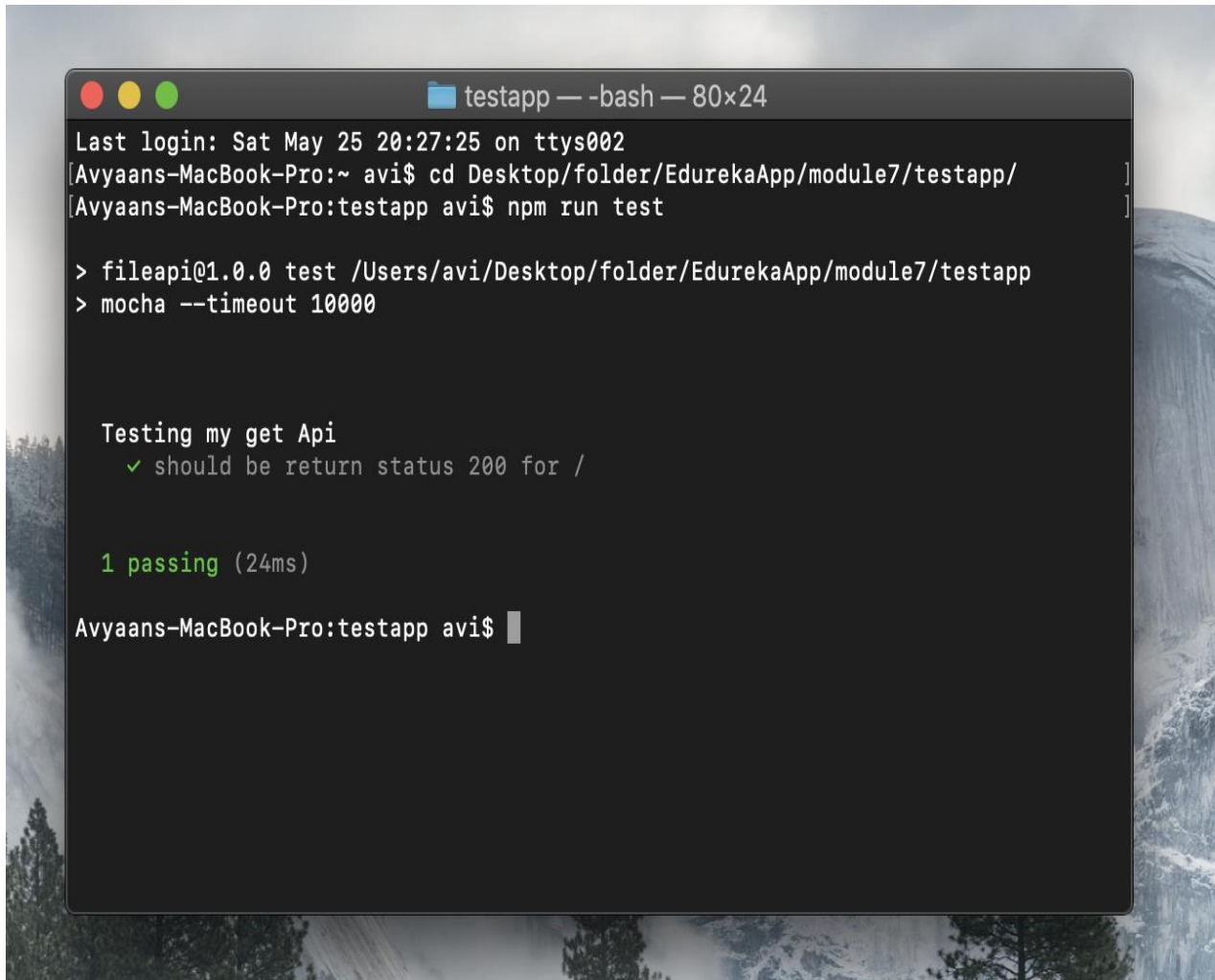
Module 9: Testing Node.js Application



The screenshot shows the VS Code editor interface. The Explorer sidebar on the left displays the project structure, including files like app.js, package.json, common.test.js, start.js, .babelrc, myText.txt, myText1.txt, db.json, and a test directory. The main editor area shows the content of common.test.js, which includes imports for 'chai' and 'chai-http', and a test case using 'describe' and 'it' to verify that a GET request to 'http://localhost:6500/' returns a status of 200. The status bar at the bottom indicates the current file is common.test.js, line 19, column 5, with 4 spaces, UTF-8 encoding, and LF line endings.

```
1 let chai = require('chai');
2 let chaiHttp = require('chai-http');
3 let expect = chai.expect;
4 chai.use(chaiHttp);
5
6 describe('Testing my get Api', () => {
7   it('should be return status 200 for /', function(done){
8     chai
9       .request('http://localhost:6500')
10      .get('/')
11      .then(function(res){
12        expect(res).to.have.status(200);
13        done();
14      })
15      .catch(function(err){
16        throw(err);
17      });
18   });
19 }
20 }
```

Step 5: To run the test case we need to execute the test command “npm test” and mocha as test runner, list all the test cases with description. It will display whether the Test case passes or fails with color indication.

A terminal window titled 'testapp — -bash — 80x24' is shown against a blurred background of a mountain landscape. The terminal displays the following text:

```
Last login: Sat May 25 20:27:25 on ttys002
[Avyaans-MacBook-Pro:~ avi$ cd Desktop/folder/EdurekaApp/module7/testapp/
[Avyaans-MacBook-Pro:testapp avi$ npm run test

> fileapi@1.0.0 test /Users/avi/Desktop/folder/EdurekaApp/module7/testapp
> mocha --timeout 10000

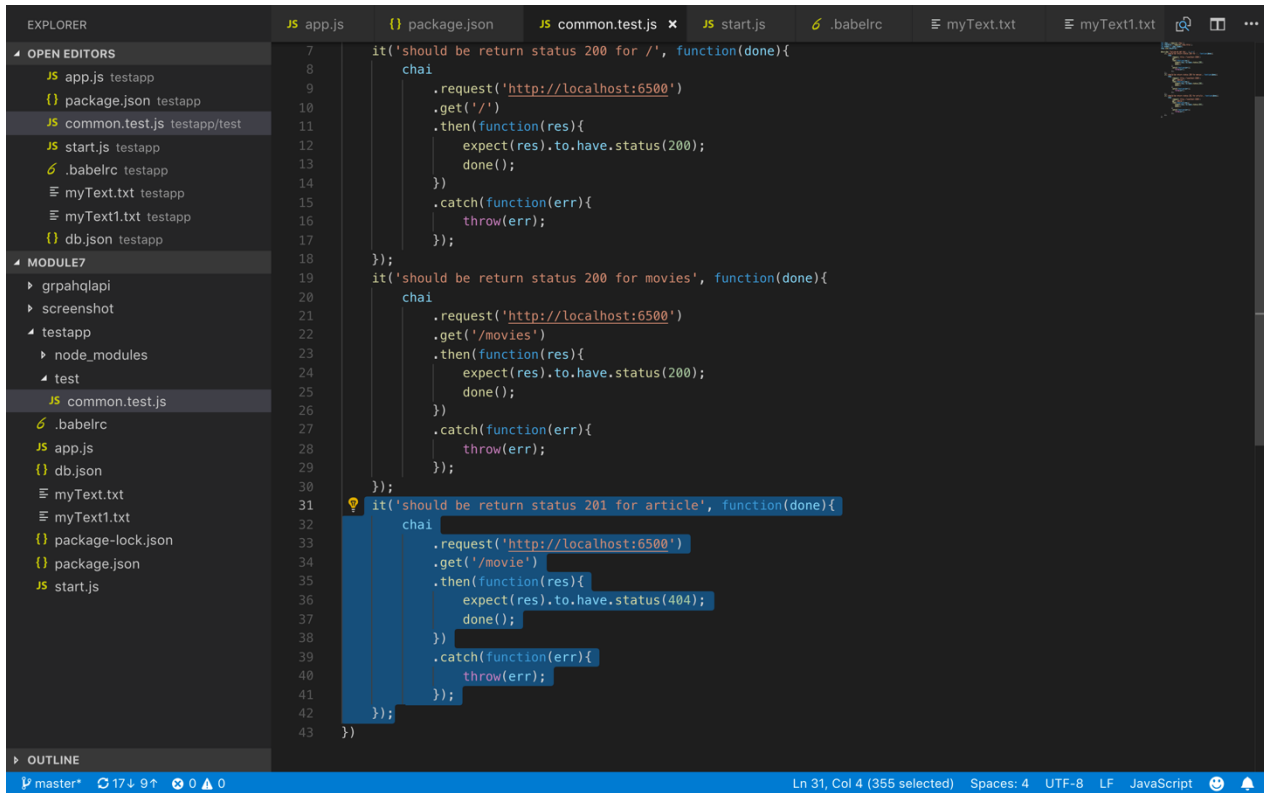
Testing my get Api
  ✓ should be return status 200 for /

1 passing (24ms)

Avyaans-MacBook-Pro:testapp avi$
```

Step 6: Similar to test for status 200 Ok, we should always write test case for status 404 in Line 31, write test case with “it” as assertion with expected response as 404 as well as catch also.

Module 9: Testing Node.js Application



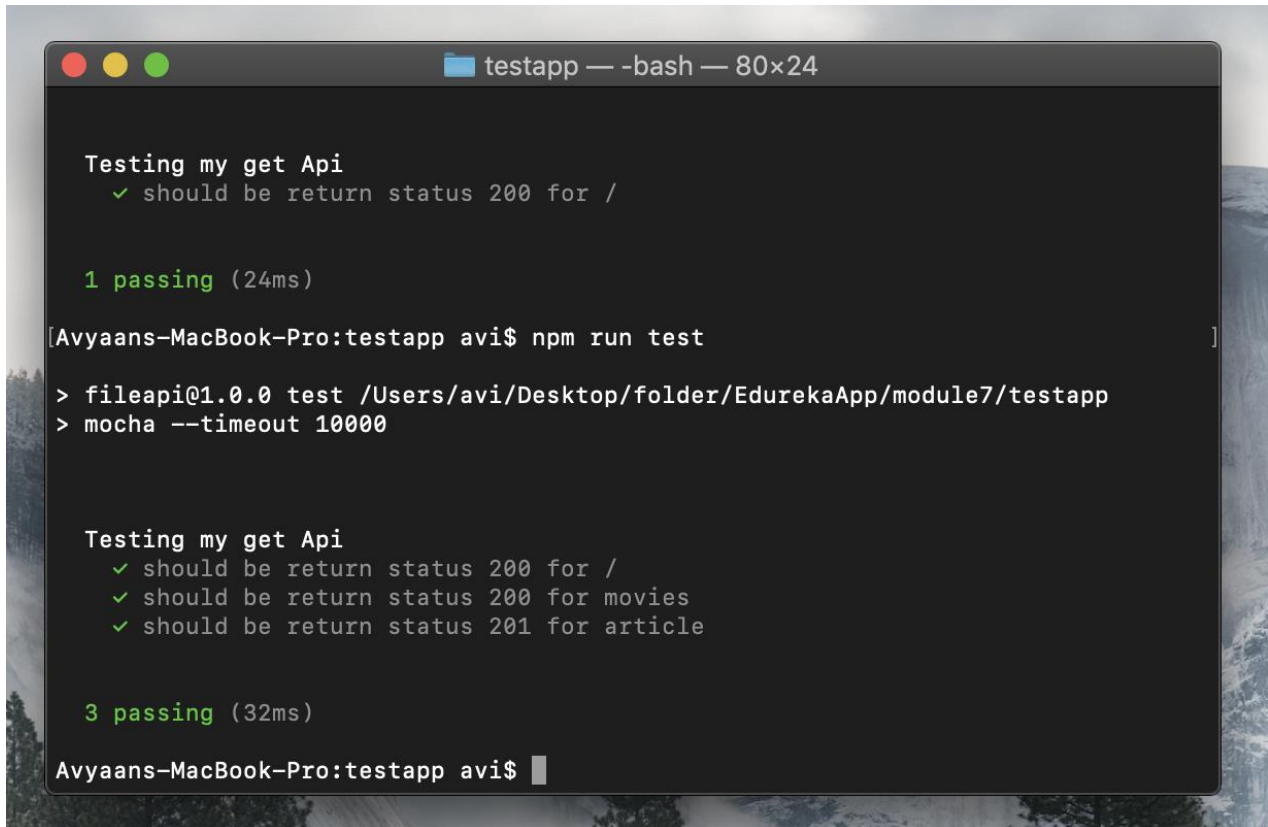
The screenshot shows a VS Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with files like app.js, package.json, common.test.js, start.js, .babelrc, myText.txt, myText1.txt, db.json, and test. The code editor shows the content of common.test.js, which contains three test cases using the 'it' function and 'chai' assertions. The first test case checks for a status of 200 for the root path '/'. The second test case checks for a status of 200 for the path '/movies'. The third test case checks for a status of 201 for the path '/movie'. The code is as follows:

```
7 it('should be return status 200 for /', function(done){
8   chai
9     .request('http://localhost:6500')
10    .get('/')
11    .then(function(res){
12      expect(res).to.have.status(200);
13      done();
14    })
15    .catch(function(err){
16      throw(err);
17    });
18 });
19
20 it('should be return status 200 for movies', function(done){
21   chai
22     .request('http://localhost:6500')
23     .get('/movies')
24     .then(function(res){
25       expect(res).to.have.status(200);
26       done();
27     })
28     .catch(function(err){
29       throw(err);
30     });
31 });
32
33 it('should be return status 201 for article', function(done){
34   chai
35     .request('http://localhost:6500')
36     .get('/movie')
37     .then(function(res){
38       expect(res).to.have.status(404);
39       done();
40     })
41     .catch(function(err){
42       throw(err);
43     });
44 });
```

Step 7: Finally when we run all the test case using mocha ,we get all the test case pass if in case API responded as expected

On this test we get all response with description

In this case all test case passed with positive response.



A terminal window titled 'testapp — -bash — 80x24' showing the execution of a test suite. The first test run shows one passing test. The second test run, initiated by 'npm run test', shows three passing tests. The terminal output is as follows:

```
Testing my get Api
  ✓ should be return status 200 for /

1 passing (24ms)

[Avyaans-MacBook-Pro:testapp avi$ npm run test]

> fileapi@1.0.0 test /Users/avi/Desktop/folder/EdurekaApp/module7/testapp
> mocha --timeout 10000

Testing my get Api
  ✓ should be return status 200 for /
  ✓ should be return status 200 for movies
  ✓ should be return status 201 for article

3 passing (32ms)

Avyaans-MacBook-Pro:testapp avi$
```

edureka!