Lesson 11 Demo 05

Testing a Promise

Objective: To test a Node.js app with a Promise object using Mocha for robust validation and verification of asynchronous functionality and error handling

Tools Required: Visual Studio

Prerequisites: Knowledge of JavaScript, Promise concept, and Node.js

Steps to be followed:

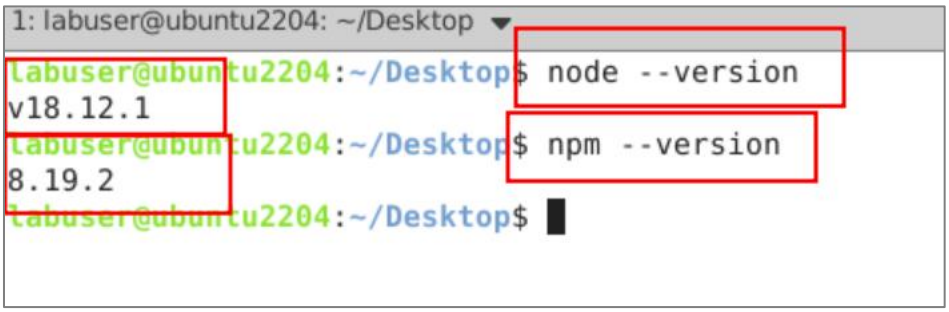
1. Check whether Node.js is installed or not
2. Create the project structure, package.json file and install Mocha module
3. Create Node.js app which returns the promise data with fake REST API program and test file to check the promise

Step 1: Check whether Node.js is installed or not

1.1 Verify whether the node is installed successfully by executing the following commands:

node --version
npm --version

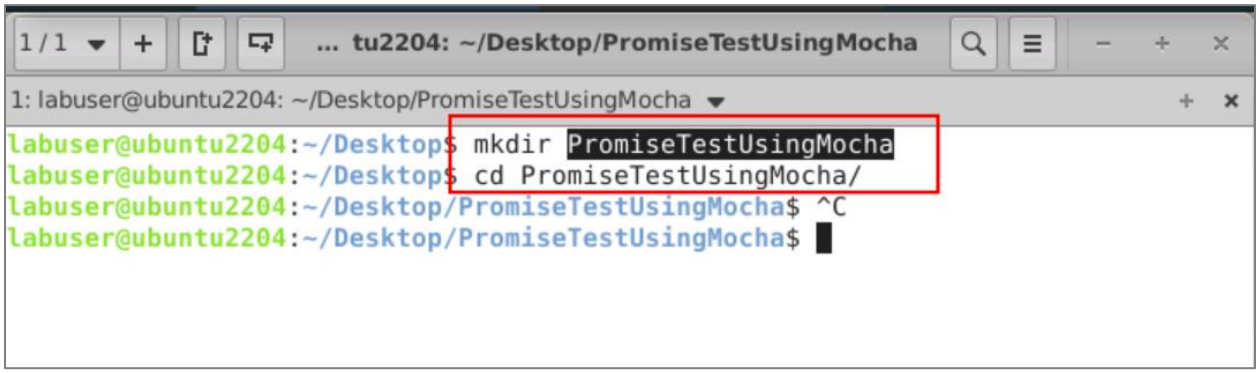
If node installation is successful, the following output will appear:



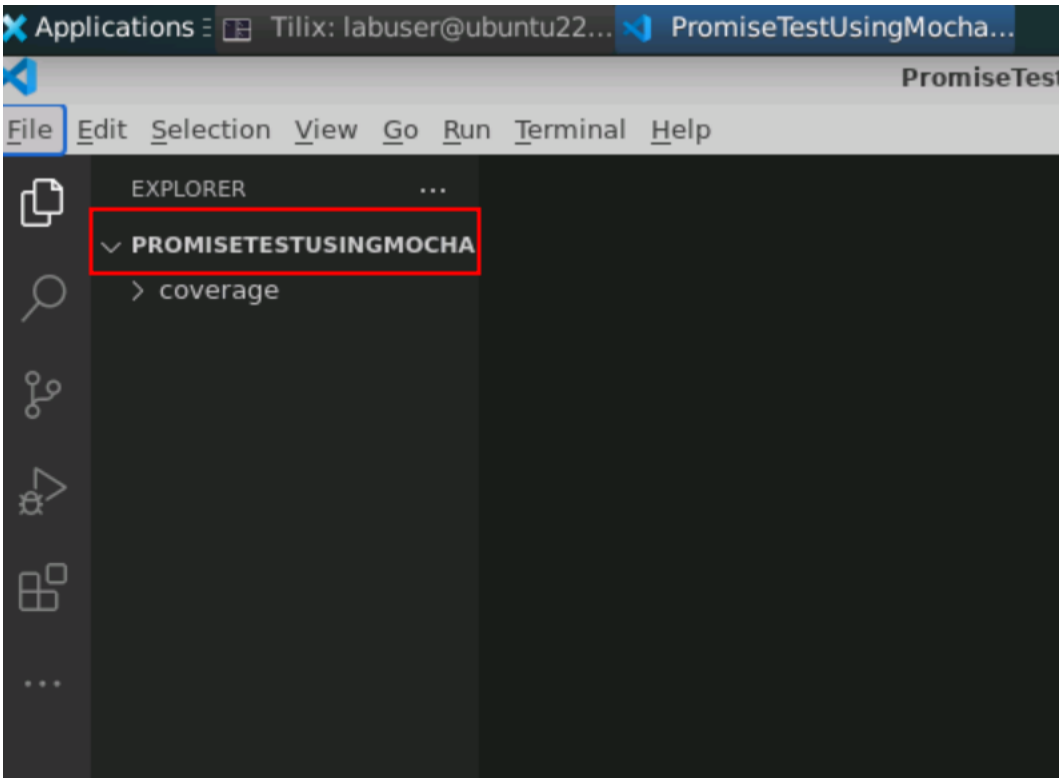
Step 2: Create the project structure, package.json file and install Mocha module

2.1 Run the following code to create a directory to hold the application and make it the working directory:

mkdir PromiseTestUsingMocha
cd PromiseTestUsingMocha



2.2 Open the following folder in the VSCode IDE as shown in the screenshot below:



2.3 Open the terminal and run the following command to create a **package.json** file for the application:

npm init

```
1/1 ... tu2204: ~/Desktop/PromiseTestUsingMocha
1: labuser@ubuntu2204: ~/Desktop/PromiseTestUsingMocha
labuser@ubuntu2204:~/Desktop/PromiseTestUsingMocha$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help init` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (promisetestusingmocha)
```

w.

Then, at last, enter yes

```
1/1 ... tu2204: ~/Desktop/PromiseTestUsingMocha
1: labuser@ubuntu2204: ~/Desktop/PromiseTestUsingMocha
entry point: (index.js)
test command:
git repository:
keywords:
author:
license: (ISC)
About to write to /home/labuser/Desktop/PromiseTestUsingMocha/package.json:

{
  "name": "promisetestusingmocha",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC"
}

Is this OK? (yes) yes
```

```
{} package.json ×
{} package.json > ...
1  {
2    "name": "nodejstestusingmocha",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
       ▷ Debug
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC"
11 }
12
```

2.4 Run **npm install mocha -D** command to install Mocha

```
1: labuser@ubuntu2204: ~/Desktop/PromiseTestUsingMocha
labuser@ubuntu2204:~/Desktop/PromiseTestUsingMocha$ npm install mocha -D

added 77 packages, and audited 78 packages in 2s

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

found 0 vulnerabilities
labuser@ubuntu2204:~/Desktop/PromiseTestUsingMocha$
```

You can view these dependencies in the package.json file

```
{} package.json ×
{} package.json > {} devDependencies > mocha
1  {
2    "name": "nodejstestusingmocha",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
       Debug
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1"
8    },
9    "author": "",
10   "license": "ISC",
11   "devDependencies": {
12     "mocha": "^10.2.0"
13   }
14 }
```

Step 3: Create Node.js app which returns the promise data with fake REST API program and test file to check the promise

3.1 Create a src folder which contains all the source code
Inside the src folder, create api.js
Inside the src folder, create app.js

3.2 Create a test folder which contains all the source code
Inside the test folder, create appTest.js

```
EXPLORER ...   {} package.json   JS api.js   JS app.js   JS appTest.js ×
PROMISETESTUSINGMOCHA          test > JS appTest.js
  > coverage                     1
  > node_modules
  v src
    JS api.js
    JS app.js
  v test
    JS appTest.js
  {} package-lock.json
  {} package.json
```

3.3 Make the following changes in the package.json file:

```
package.json ×   JS api.js   JS app.js   JS appTest.js
{} package.json > {} devDependencies
1   {
2     "name": "promisetestusingmocha",
3     "version": "1.0.0",
4     "description": "",
5     "main": "index.js",
      ▷ Debug
6     "scripts": {
7       "test": "mocha test/*.js"
8     },
9     "author": "",
10    "license": "ISC",
11    "devDependencies": {
12      "mocha": "^10.2.0"
13    }
14  }
15
```

3.4 Create an api.js file with the following code

```
JS api.js > [e] <unknown> > price
module.exports = [
    {
        pid:1,
        pname:"Tv",
        price:55000
    },
    {
        pid:2,
        pname:"Laptop",
        price:87000
    },
    {
        pid:3,
        pname:"Computer",
        price:42000
    }
]
```

3.5 Create an app.js file with the following code

```
package.json   api.js   app.js   appTest.js
src > app.js > findProduct
const products = require("./api");
function findProduct(productid){
    return new Promise((resolve,reject)=> {
        setTimeout(()=> {
            const filterProduct = products.find(p=>p.pid=productid);
            filterProduct
            ? resolve({message:"Product found",result:filterProduct})
            : reject(new Error("Product not found"))
        },1000)
    })
}
module.exports = {findProduct}
```

3.6 Write the code to test promise in appTest.js

```
appTest.js > describe("Promise Fake Test API") callback
const assert = require("assert");
const appRef = require("../src/app");
describe("Promise Fake Test API",()=> {
    it("Product found test ", async ()=> {
            const actulaResult  = await appRef.findProduct(1);
            const expectResult = {
                "message":"Product found",
                "result":{
                    pid:1,
                    pname:"Tv",
                    price:55000
                }
            }
            assert.deepEqual(actulaResult,expectResult);
    })

    it("Product not found test ",async()=> {
        try{
        const actulaResult  = await appRef.findProduct(5);
        }catch(error){
            assert.equal(error.message,"Product not found");
        }
    })

})
```

3.7 Execute the following command in the **terminal** to run the file:

```
1: labuser@ubuntu2204: ~/Desktop/PromiseTestUsingMocha
labuser@ubuntu2204:~/Desktop/PromiseTestUsingMocha$ npm test

> promisetestusingmocha@1.0.0 test
> mocha test/*.js


  Promise Fake Test API
    ✔ Product found test  (1004ms)
    ✔ Product not found test  (1000ms)


  2 passing (2s)

labuser@ubuntu2204:~/Desktop/PromiseTestUsingMocha$
```

Here npm test checks the package.json file and you have already provided all testing file details in test mocha test/*.js

```
package.json ×   api.js   app.js   appTest.js
package.json > {} devDependencies
1   {
2     "name": "promisetestusingmocha",
3     "version": "1.0.0",
4     "description": "",
5     "main": "index.js",
      Debug
6     "scripts": {
7       "test": "mocha test/*.js"
8     },
9     "author": "",
10    "license": "ISC",
11    "devDependencies": {
12      "mocha": "^10.2.0"
13    }
14  }
15
```

Here both tests are passed when you input the correct product id as well as wrong id.

By following these steps, you have successfully tested a Node.js app with a Promise object using Mocha to ensure thorough validation of asynchronous functionality and effective error handling.