# Learning Outcome

# After completing this module, the student should be Able to develop the real time scenarios based on Node JS applications.

To meet the learning outcome, a student has to complete the following activities

1. Create Module (Function) & export, import using Node
2. Create a Web Application using Express JS
3. Create RestAPI using express JS

# Activity 1

## Aim: Create Module (Function) & export, import using Node

**Learning outcome:**Able to develop the real time scenarios based on Node JS applications.

**Duration:** 3 Hour

**List of Hardware/Software requirements:**

1. Operating System – Windows 10/11 or Linux
2. Command Prompt/Power Shell
3. Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
4. NodeJS

**Code/Program/Procedure (with comments):**

* + - 1. Creating a Module: Modules are created in Node.js are JavaScript files. Every time a new file with .js extension is created, it becomes a module.
      2. Create a file func.js

function add (x, y) {

return x + y;

}

function subtract (x, y) {

return x - y;

}

// Adding the code below to allow importing

// the functions in other files

module.exports = { add }

* + - 1. Create a file **main.js** which will import the **func.js** module.

// Importing the func.js module

// The ./ says that the func module is in the same directory as the main.js file

const f = require('./func');

// Require returns an object with add() and stores it in the f variable

// which is used to invoke the required

const result = f.add(10, 5);

console.log ('The result is:', result);

* + - 1. Run the code using **node main.js**



**References:**

* <https://www.geeksforgeeks.org/import-and-export-in-node-js/?ref=rp>
* <https://javascript.plainenglish.io/create-a-single-page-website-using-node-js-and-express-js-a0b53e396e4f>

# Activity 2

## Aim: Create a Web Application using Express JS

**Learning outcome:**Able to develop the real time scenarios based on Node JS applications.

**Duration:** 4 Hour

**List of Hardware/Software requirements:**

1. Operating System – Windows 10/11 or Linux
2. Command Prompt/Power Shell
3. Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
4. NodeJS
5. Express JS

**Code/Program/Procedure (with comments):**

* + - 1. Make a file app.js usingVisual Studio Code.

var express = require('express');

var app = express();

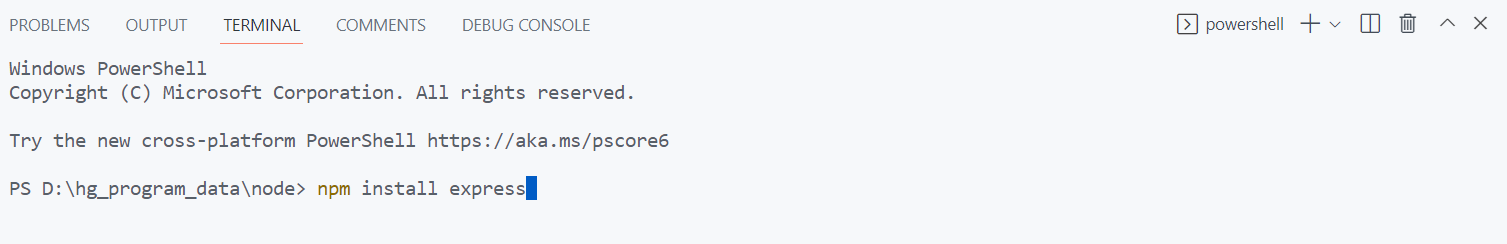
// define routes here..

var server = app.listen(5000, function () {

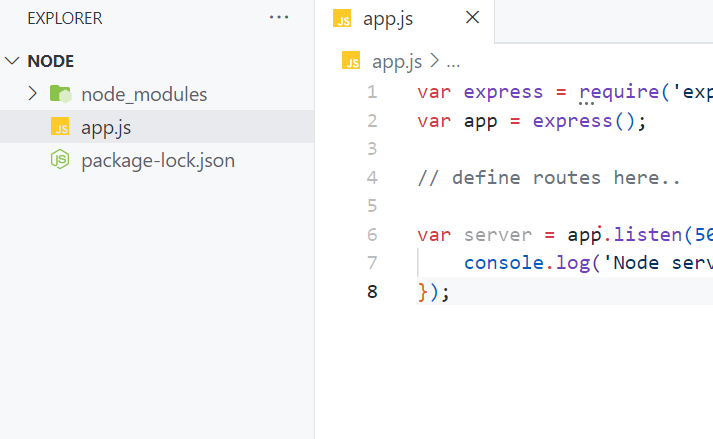
console.log('Node server is running..');

});

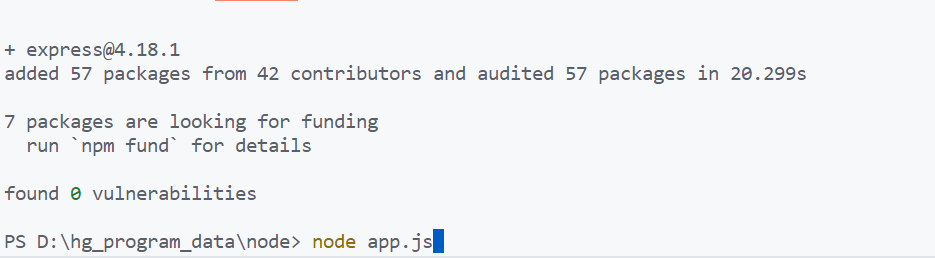
* + - 1. Open terminal in that specific folder to install all modules as mentioned below

****

* + - 1. Now we can see all required modules are installed.



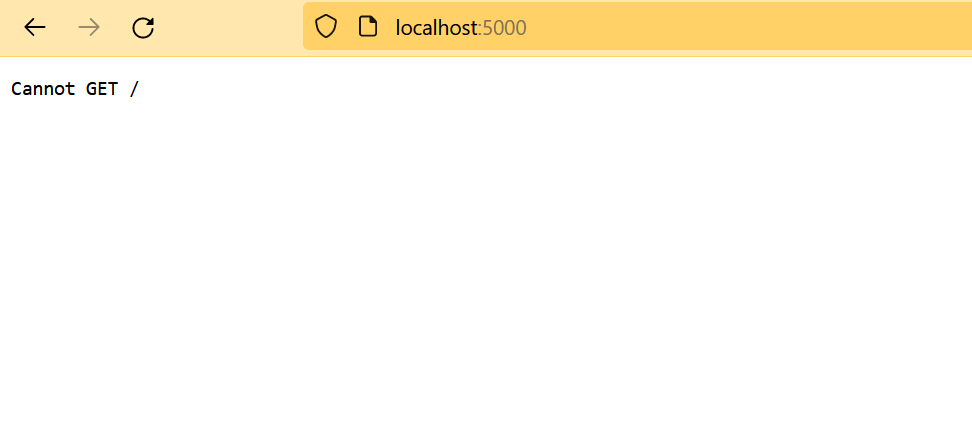
* + - 1. Now run the app using **node app.js**



* + - 1. Now app is running on local system on port address 5000 and the link is localhost:5000



* + - 1. check link on browser



* + - 1. We are not able to see any output because routes are not defined yet. Modify the **app.js** with following code:

var express = require('express');

var app = express();

app.get('/', function (req, res) {

res.send('<html><body><h1>Hello World</h1></body></html>');

});

app.post('/submit-data', function (req, res) {

res.send('POST Request');

});

app.put('/update-data', function (req, res) {

res.send('PUT Request');

});

app.delete('/delete-data', function (req, res) {

res.send('DELETE Request');

});

var server = app.listen(5000, function () {

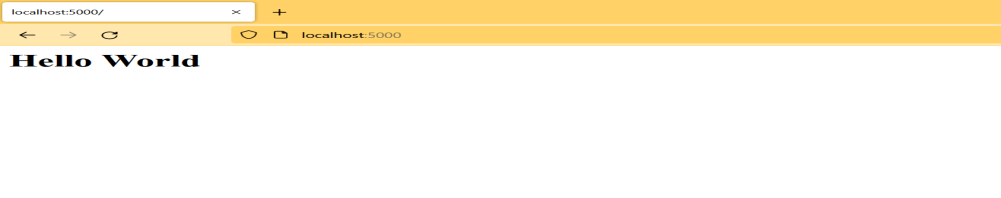
console.log('Node server is running..');

});

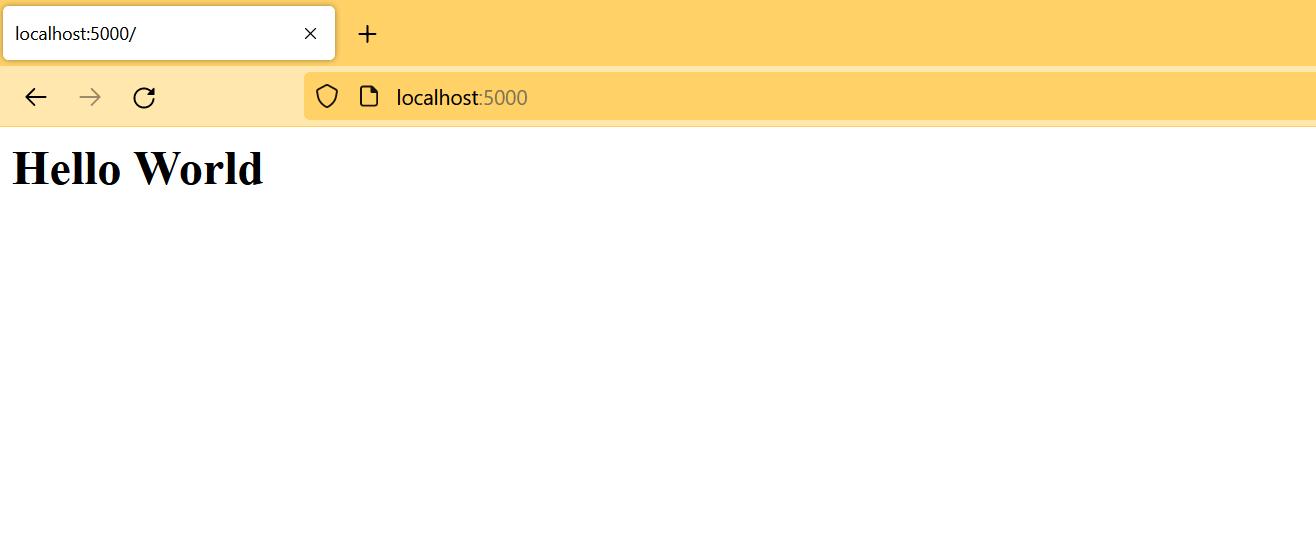
* + - 1. Now run the file again using node app.js command.



* + - 1. Server running on the same port address with the message “Hello World”



**Output/Results snippet:**

****

**References:**

* [**https://www.tutorialsteacher.com/nodejs/expressjs-web-application**](https://www.tutorialsteacher.com/nodejs/expressjs-web-application)

# Activity 3

## Aim: Create RestAPI using express JS

**Learning outcome:**Able to develop the real time scenarios based on Node JS applications.

**Duration:** 4 Hour

**List of Hardware/Software requirements:**

1. Operating System – Windows 10/11 or Linux
2. Command Prompt/Power Shell
3. Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
4. NodeJS
5. Express JS

**Code/Program/Procedure (with comments):**

* + - 1. Make a file name **index.js** using visual studio code.

import express from 'express';

import bodyParser from 'body-parser';

import users Routes from './routes/users.js'

constapp=express();

constPORT=5000;

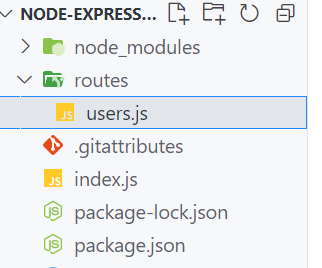
app.use(bodyParser.json());

app.use('/users', usersRoutes);

app.get('/', (req, res)=>res.send('This is Homepage, kindly refer json file here http://localhost:5000/users'));

app.listen(PORT, (req, res)=>console.log(`Server is Running on port [http://localhost:${PORT}`)](http://localhost:$%7bPORT%7d%60)));

* + - 1. Install all required module with the help of **npm install express**



* + - 1. Make routes folder and create users.js in the same folder.

import express from'express'; // Web Framework for nodejs

import { v4 as uuidv4 } from'uuid'; //Get unique id every time

const router =express.Router(); //using express for routing

let users = [

{

"firstName": "John",

"lastName": "Doe",

"age": 25,

"id": "fcb4efa5-608d-4d91-aed3-33af404928ae"

},

{

"firstName": "Johny",

"lastName": "Doe",

"age": 25,

"id": "2fb6fed6-43f3-4ee6-ab04-2bd3782c1886"

},

{

"firstName": "Jason",

"lastName": "Dew",

"age": 30,

"id": "a4f12aa0-8fd5-473d-9834-aa98a2bdd84c"

},

{

"firstName": "BSon",

"lastName": "DB",

"age": 40,

"id": "179c592c-1d4d-451c-808c-e72749e9c880"

}

] //variable used to store data

// GET METHOD i.e. Read request can done from browser

router.get('/', (req, res)=>{

// console.log(users);

res.send(users);

});

// POST METHOD i.e. CREATE request can be done from POSTMAN/THUNDER CLIENT

router.post('/', (req, res)=>{

// console.log('POST ROUTE Reached');

// console.log(req.body);

constuser=req.body; //when post request implemented, data stored in body and stored in variable i.e. 'user'

constuserID=uuidv4(); // For unique id⇨ '9b1deb4d-3b7d-4bad-9bdd-2b0d7b3dcb6d'

constuserWithID= { ...user, id: userID } //Spread Operator

users.push(userWithID); //With Push Function it will add data in "users" array

// res.send('POST ROUTE Reached');

res.send(`User with the name ${user.firstName} added to the database.`); // Confirmation message on client side

});

// GET METHOD via passing id, Read request can done from browser

router.get('/:id', (req, res)=>{

// console.log(req.params);

// res.send('THE GET ID ROUTE');

// res.send(req.params);

const { id } =req.params; //id will store in params and stored in variable

constfoundUser=users.find((user)=>user.id ==id); //using find filter to fetch exact same id

res.send(foundUser);

});

// DELETE METHOD via passing id i.e. DELETE request can be done from POSTMAN/THUNDER CLIENT

router.delete('/:id', (req, res)=>{

const { id } =req.params;

users=users.filter((user)=>user.id !==id);

res.send(`User with the id ${id} deleted from the database`);

});

// PATCH METHOD(PARTIAL MODIFICATION) request via passing id i.e. UPDATE request can be done from POSTMAN/THUNDER CLIENT

router.patch('/:id', (req, res)=>{

const { id }=req.params;

const { firstName, lastName, age } =req.body;

constuser=users.find((user)=>user.id ===id);

if (firstName) {

user.firstName=firstName;

}

if (lastName) {

user.lastName=lastName;

}

if (age) {

user.age=age;

}

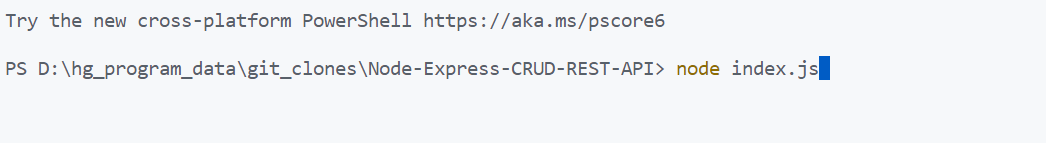
res.send(`User with the id ${id} has been updated`);

});

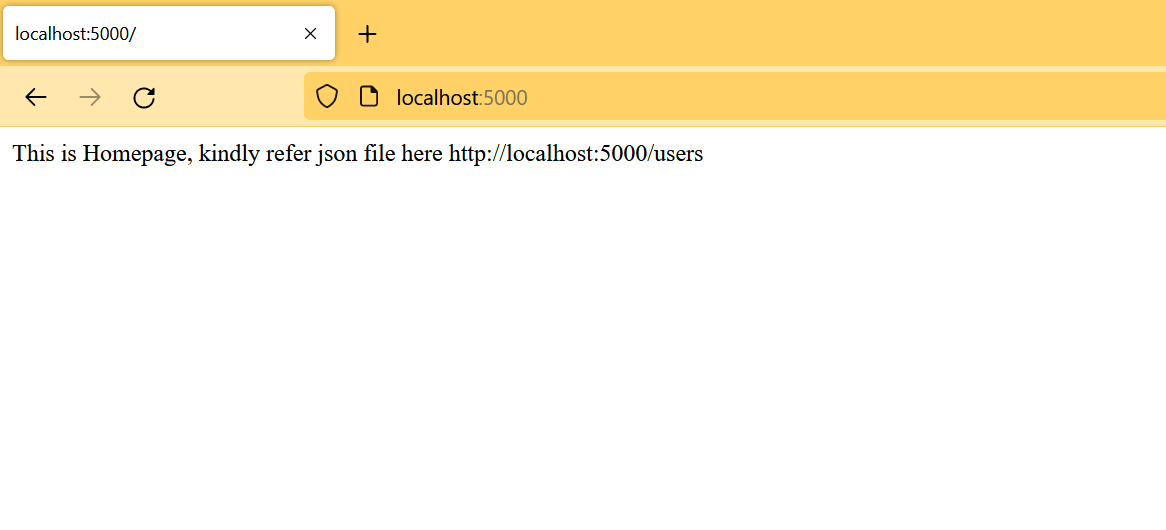
// Note - PUT method is used to completely overwrite

exportdefaultrouter;

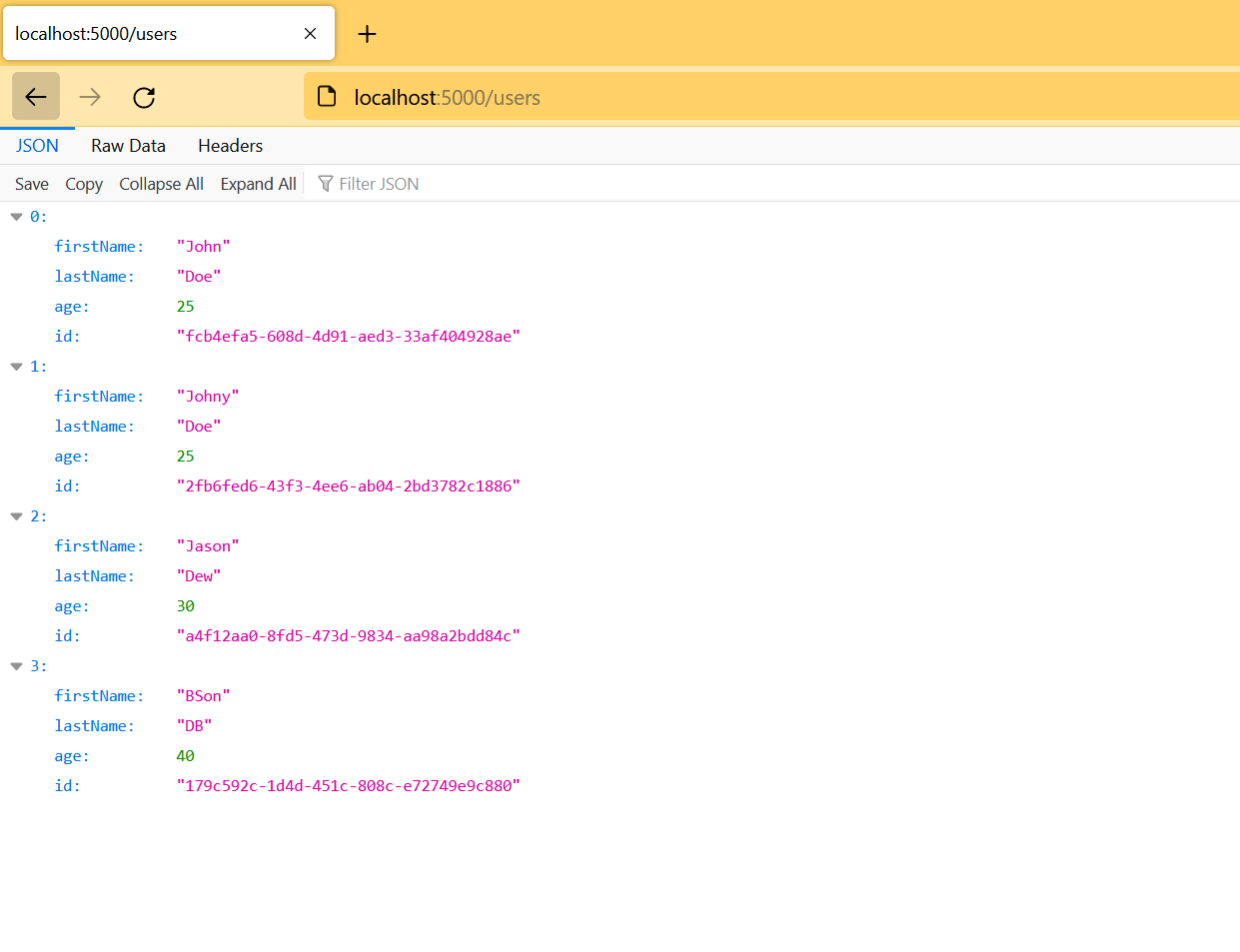
* + - 1. Now run the application using **node index.js**



* + - 1. Referring localhost:5000/users to see data that is mentioned in JSON format.



* + - 1. The data available is JSON format.



**References:**

* <https://www.tutorialsteacher.com/nodejs/expressjs>