## Learning Outcome 1

## Activity 2

**Aim:** Create an Angular application which add, multiply, divide the two input numbers from textbox.

# **Learning outcome**: Able to understand how to create an angular application.

###### Duration: 3 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

1. Open vscode and create a .html file.

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title></title>  </head>  <body ng-app>  Insert your angular code here.  </body>  </html> |

Addition code input.

|  |
| --- |
| <h4>Enter Numbers to Add: </h4>  <!-- First row of inputs -->  <div style="display: flex; " class="m-2">  <div><input type="number" class="form-control col-3" ng-model="num1"></div>  <span><strong style="font-size: x-large; color: red; margin-left: 5px; margin-right: 5px;">+</strong></span>  <div><input type="number" class="form-control col-3" ng-model="num2"></div>  <span><strong style="color: green; font-size: larger; margin-left: 10px;" ng-bind="num1+num2"></strong></span>  </div> |

Subtract code input.

|  |
| --- |
| <h4>Enter Number to Subtract:</h4>  <div style="display: flex;"class="m-2">  <div><input type="number" class="form-control col-3" ng-model="num3"></div>  <span><strong style="font-size: x-large; color: red; margin-left: 5px; margin-right: 5px;">-</strong></span>  <div><input type="number" class="form-control col-3" ng-model="num4"></div>  <span><strong style="color: green; font-size: larger; margin-left: 10px;" ng-bind="num3-num4"></strong></span>  </div> |

Multiply code input.

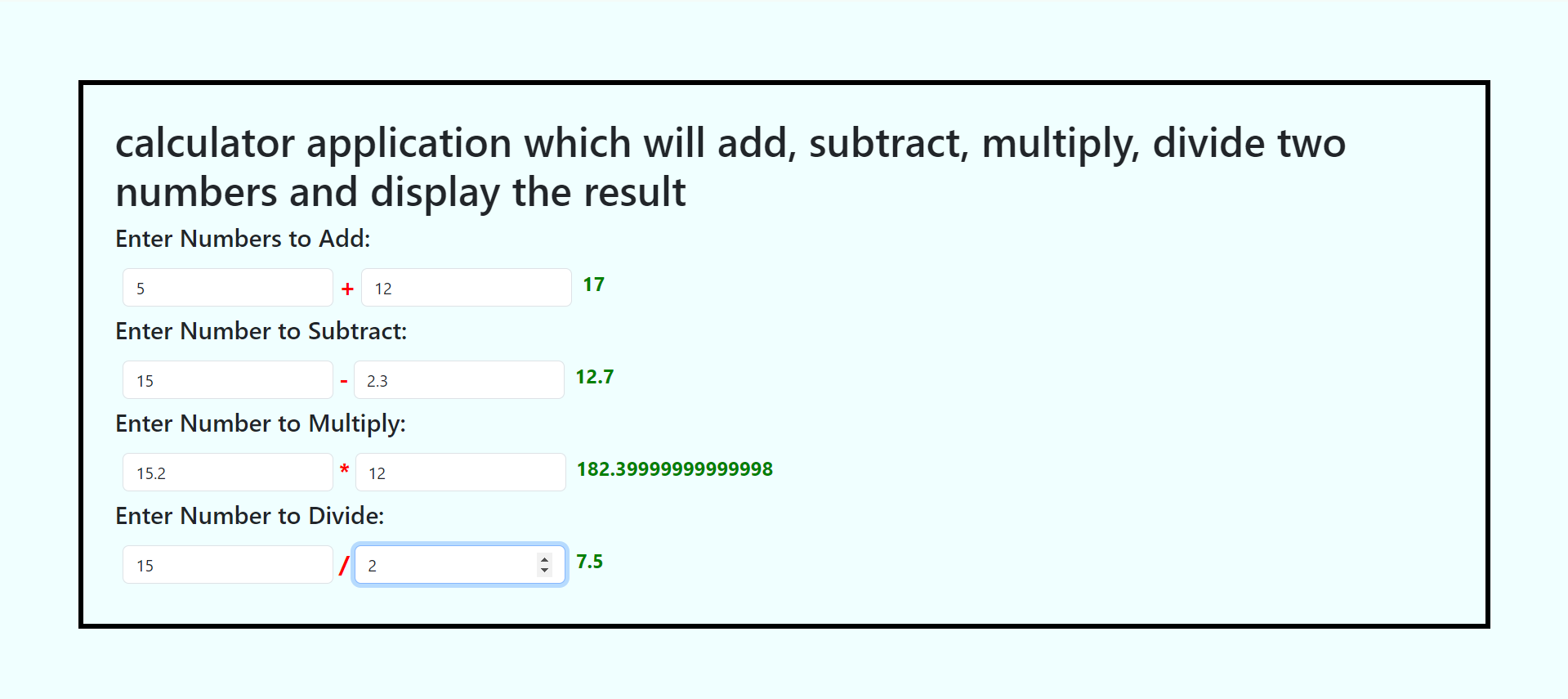
|  |
| --- |
| <h4>Enter Number to Multiply:</h4>  <div style="display: flex;"class="m-2">  <div><input type="number" class="form-control col-3" ng-model="num5"></div>  <span><strong style="font-size: x-large; color: red; margin-left: 5px; margin-right: 5px;">\*</strong></span>  <div><input type="number" class="form-control col-3" ng-model="num6"></div>  <span><strong style="color: green; font-size: larger; margin-left: 10px;" ng-bind="num5\*num6"></strong></span>  </div> |

Divide code input.

|  |
| --- |
| <h4>Enter Number to Divide:</h4>  <div style="display: flex;"class="m-2">  <div><input type="number" class="form-control col-3" ng-model="num7"></div>  <span><strong style="font-size: x-large; color: red; margin-left: 5px; margin-right: 5px;">/</strong></span>  <div><input type="number" class="form-control col-3" ng-model="num8"></div>  <span><strong style="color: green; font-size: larger; margin-left: 10px;" ng-bind="num7/num8"></strong></span>  </div> |

**Output:**

Open file on any browser.



## Learning Outcome 2

## Activity 1

**Aim:** Create an Angular application which can validate the email accepted from user.

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

###### Duration: 3 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

1. Open the vscode and go to terminal and check the node and npm version.

**node -v**

**npm -v**

1. Then install the angular with this command.

**npm install -g @angular/cli**

1. After that open the cmd and type this command and check the angular version for this command.

**ng version**



1. I have created a folder where I want to create a new app. Click on terminal and run this command. test is my folder name.

**cd test**

**ng new validation –no-standalone**

after that’s wait for some time.

1. Open the **src\app\app.module.ts** and insert some line of codes.

Input:

|  |
| --- |
| import { FormsModule, ReactiveFormsModule } from '@angular/forms';  @NgModule({  declarations: [  AppComponent  ],  imports: [  BrowserModule,  AppRoutingModule,  FormsModule, //also add here what we have import  ReactiveFormsModule //also add here what we have import  ],  providers: [  provideClientHydration()  ],  bootstrap: [AppComponent]  })  export class AppModule { } |

1. Now open the **src\app\app.component.html** and insert some line of code.

Input:

|  |
| --- |
| <h1>Email Validation</h1>  <form [formGroup]="form" (ngSubmit)="onSubmit()">    <div class="mb-3">  <label for="" class="form-label">Email</label>  <input  type="email"  class="form-control"  name=""  id=""  formControlName="email" pattern="[a-z0-9.\_%+-]+@[a-z0-9.\_%+-]+\.[a-z]{2,3}$" required  aria-describedby="emailHelpId"  placeholder="abc@mail.com"  />    </div>  <div \*ngIf="f['email'].touched && f['email'].invalid" class="alert alert-danger">  <div \*ngIf="f['email'].errors && f['email'].errors['required']">Email Required</div>  <div \*ngIf="f['email'].errors && f['email'].errors['pattern']">Please, enter valid email address.</div>  </div>  <button type="submit" class="btn btn-primary">Submit</button>  </form> |

1. Now open the **src\app\app.component.ts** and insert some line of code.

Input:

|  |
| --- |
| import { Validators, FormGroup, FormControl } from '@angular/forms';  form = new FormGroup({  email: new FormControl('', [Validators.required, Validators.email])  });  get f() { return this.form.controls; } // <-- GETTERS -->  onSubmit() {  if (this.form.valid) {  alert(`Email submitted: ${this.f.email.value}`);  console.log(`Email submitted: ${this.f.email.value}`);  } else {  console.log('Form is invalid');  }  }; |

1. We need to install the bootstrap.

**npm install bootstrap –save**

1. Now follow this setup to copy the bootstrap relative path.

* Go to the node\_modules
* Then bootstrap
* Then dist
* Then css
* Find a bootstrap.min.css file, right click on it and copy the relative path like below.

test\validation\node\_modules\bootstrap\dist\css\bootstrap.min.css

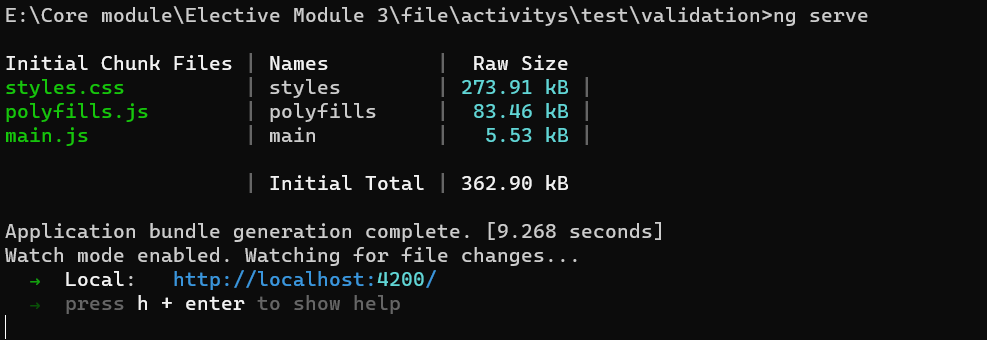
1. Now go to this folder **src/styles.css** and insert some lines of code.

**@import "bootstrap/dist/css/bootstrap.css";** //we need to remove the .min, change this \ to this / and cut some other folder we want path start from bootstrap.

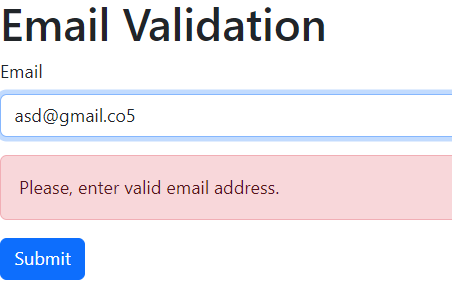
1. After these all step run this command in terminal.

**ng serve**

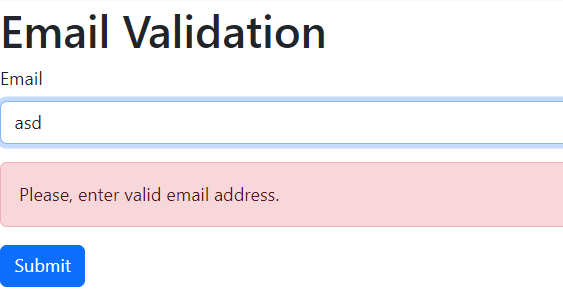
and click the line.

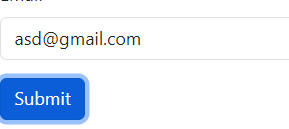


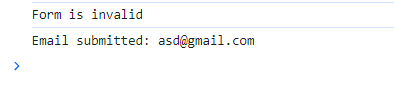
Output:



Second Image.







## Learning Outcome 3

## Activity 1

**Aim:** Create an Angular application which can create Captcha.

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

###### Duration: 5 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

Create the Angular application

1. Let's create the application with the Angular base structure using the @angular/cli with the route file and the SCSS style format.

ng new angular-recaptcha –no-standalone

1. Right-click on the created project, open the terminal, and run this command.

npm install ng-recaptcha

1. Create a re-captcha sitekey. <https://www.google.com/recaptcha/about/>
2. Add this code into the app.module.ts

|  |
| --- |
| import { NgModule } from '@angular/core';  import { BrowserModule, provideClientHydration } from '@angular/platform-browser';  import { FormsModule, ReactiveFormsModule } from '@angular/forms';  import { AppRoutingModule } from './app-routing.module';  import { AppComponent } from './app.component';  import { RecaptchaFormsModule, RecaptchaModule } from 'ng-recaptcha';  @NgModule({  declarations: [  AppComponent  ],  imports: [  BrowserModule,  AppRoutingModule,  // ////////////////////////////////  FormsModule,  ReactiveFormsModule,  RecaptchaModule,  RecaptchaFormsModule  ],  providers: [  provideClientHydration()  ],  bootstrap: [AppComponent]  })  export class AppModule { } |

1. Add this code to the app.component.html

|  |
| --- |
| <h1>Recaptcha Google</h1>  <form [formGroup]="form" >  <div class="mb-3 ">  <re-captcha formControlName="captcha"  siteKey="add your sitekey"  ></re-captcha>  <div class="alert alert-danger" \*ngIf="f['captcha'].errors && f['captcha'].errors['required']">ReCaptcha is required.</div>  </div>  <button class="btn btn-primary" type="submit">Submit</button>  </form> |

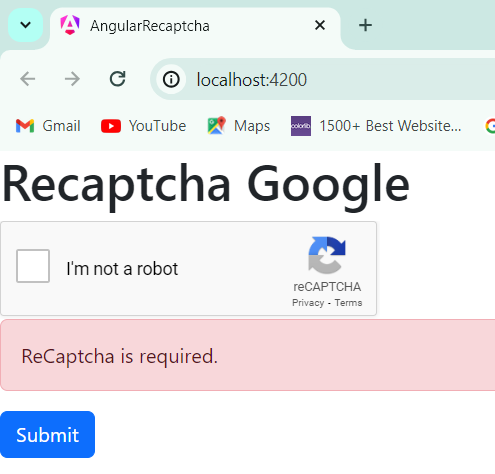
1. Add this code into the app.component.ts

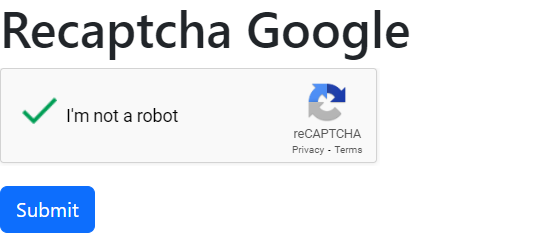
|  |
| --- |
| import { Component } from '@angular/core';  import { FormGroup,FormControl, Validators } from '@angular/forms';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrl: './app.component.css'  })  export class AppComponent {  form = new FormGroup({  captcha: new FormControl('', [Validators.required]),  });  get f( ) { return this.form.controls; }  } |

1. Now run the server.

ng serve -o

Output:





## Activity 2

**Aim:** Create Responsive Web Application using Angular.

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

###### Duration: 5 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

Create a new project.

Add this code into the app.component.html

|  |
| --- |
| <div fxLayout="column" style="height: 100vh;">  <!-- Header -->  <div class="header">  <h1>Responsive Angular App</h1>  </div>  <!-- Main Content -->  <div fxLayout="row" fxLayout.xs="column" fxLayoutAlign="start stretch" class="main-content">  <!-- Sidebar -->  <div fxFlex="20" fxFlex.xs="100" class="sidebar">  <ul>  <li><a href="#">Link 1</a></li>  <li><a href="#">Link 2</a></li>  <li><a href="#">Link 3</a></li>  </ul>  </div>  <!-- Content -->  <div fxFlex class="content">  <p>This is the main content area. It adjusts based on the screen size.</p>  </div>  </div>  </div> |

Add this code into the app.component.css

|  |
| --- |
| .header {  background-color: #333;  color: #fff;  padding: 20px;  text-align: center;  }    .main-content {  padding: 20px;  }    .sidebar {  background-color: #f4f4f4;  padding: 20px;  }    .content {  background-color: #fff;  padding: 20px;  } |

Add this line to the app.module.ts

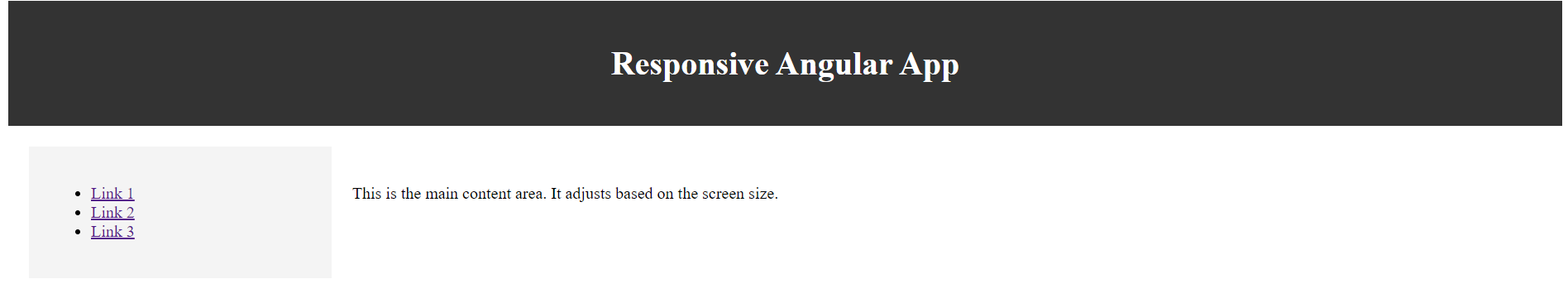
import { FlexLayoutModule } from '@angular/flex-layout';

open the terminal and run this command.

npm i @angular/flex-layout

ng serve -o

output



## Learning Outcome 4

## Activity 1

**Aim:** Create an Angular application using Nodejs with complete templating greeting system.

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

###### Duration: 2 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

1. Create a folder right-click and open the folder in the terminal and type this command.

npm install -g @angular/cli

1. Create a project.

ng new project --no-standalone

1. Now we need to generate a component with the below command. Open your project in the terminal.

ng g c greet

1. After the creation of the greet component.

Insert these code.

greet.component.ts

|  |
| --- |
| import { Component, OnInit } from '@angular/core';  @Component({  selector: 'app-greet',  templateUrl: './greet.component.html',  styleUrl: './greet.component.css'  })  export class GreetComponent implements OnInit {  ngOnInit(): void {    }  name: string= 'Padam Singh';  greet(): void{  alert(`Hello ${this.name}`);  }  } |

greet.component.html

|  |
| --- |
| <div class="p-5">  <p>greet works!</p>  <input type="text" id="name" value={{name}}>  <button type="button" (click)="greet()">Greet Me</button>  </div> |

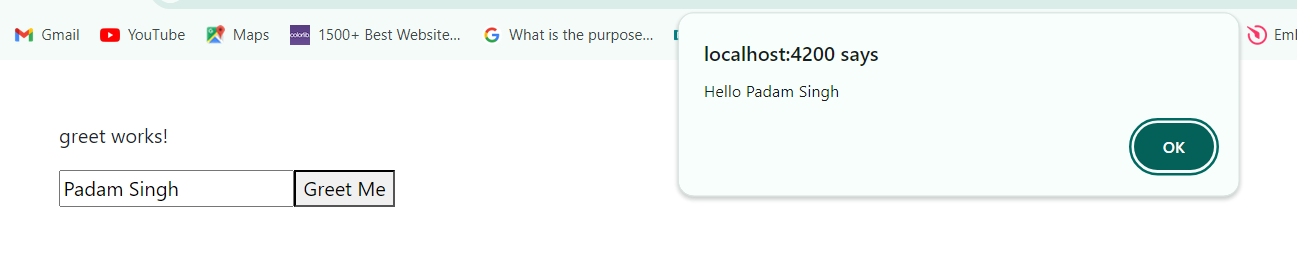
app.component.html

|  |
| --- |
| <div>  <app-greet />  </div> |

1. Run the server

ng serve -o

Output:



## Learning Outcome 4

## Activity 2

**Aim:** Create a basic Program with Node.

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

###### Duration: 2 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Internet Connectivity, Browser (Chrome or Edge)
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS, Angular CLI

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

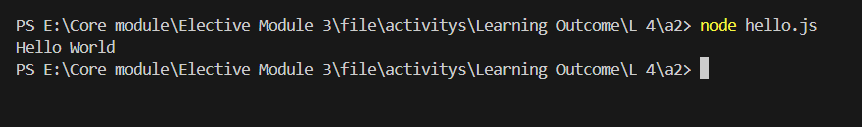
1. Create a new file hello.js
2. Insert this code.

|  |
| --- |
| Console.log(“Hello world”) |

1. Open the terminal and run this command.

node hello.js

Output:



## Learning Outcome 5

## 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 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* 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

Input:

|  |
| --- |
| 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 }; |

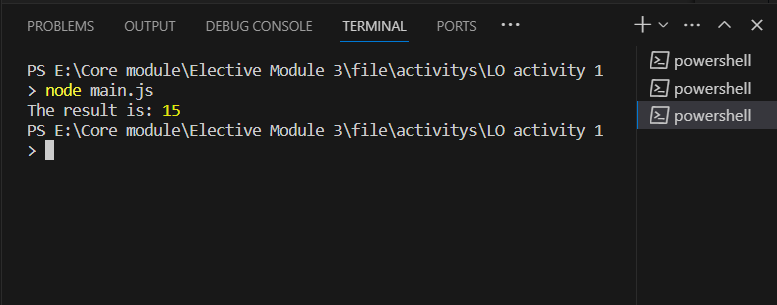
1. Create a file main.js which will import the func.js module.

Input:

|  |
| --- |
| // 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. Right click on the folder and open in terminal.
2. Run the code using **node main.js**

Output:



## 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 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS
* ExpressJS

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

1. I have a html template website.
2. Before creating a web application we need some npm packages, like express and nodemon. Now we need to install these packages.

**npm init -y**

**npm install express**

**npm install nodemon**

1. Make a file name index.js using Visual Studio Code.

Input:

|  |
| --- |
| //Importing the express  const express = require("express");  const app = express();  // Serve static files from the public directory  // We need to create a public folder for our css, js and images. this below line will be used to serve those files  app.use(express.static(\_\_dirname + '/public'));  // Create a home page route  app.get('/', function(req, res) {  // res.send("Welcome to the Home Page");  res.sendFile(\_\_dirname + "/index.html");  });  app.get('/about', function(req, res) {  res.sendFile(\_\_dirname + "/about.html");  });  app.get('/contact', function(req, res) {  res.sendFile(\_\_dirname + "/contact.html");  })  app.get('/components', function(req, res) {  res.sendFile(\_\_dirname + "/components.html");  })  app.get('/service', function(req, res) {  res.sendFile(\_\_dirname + "/services.html");  });  app.get('/project', function(req, res) {  res.sendFile(\_\_dirname + "/project.html");  });  // this below line use for server side rendering  app.listen(3000, function() {  // Log a message to the console indicating that the server is running  console.log("Server is running on: http://localhost:3000/");  }); |

After this code we need to do some little bit changes into the html pages like css, js and image.

Css

<link rel="stylesheet" type="text/css" href="/css/style.css">

js

<script type="text/javascript" src="/javascript/script.js">

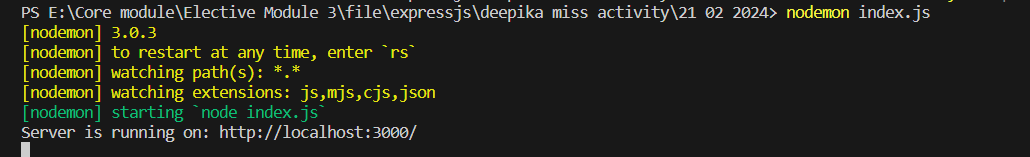
Images be like that

<img class="img-responsive" alt="" src="/assets/images/img-11.jpg">

We no need to give the path with public folder.

1. Right click on your folder and open in terminal.

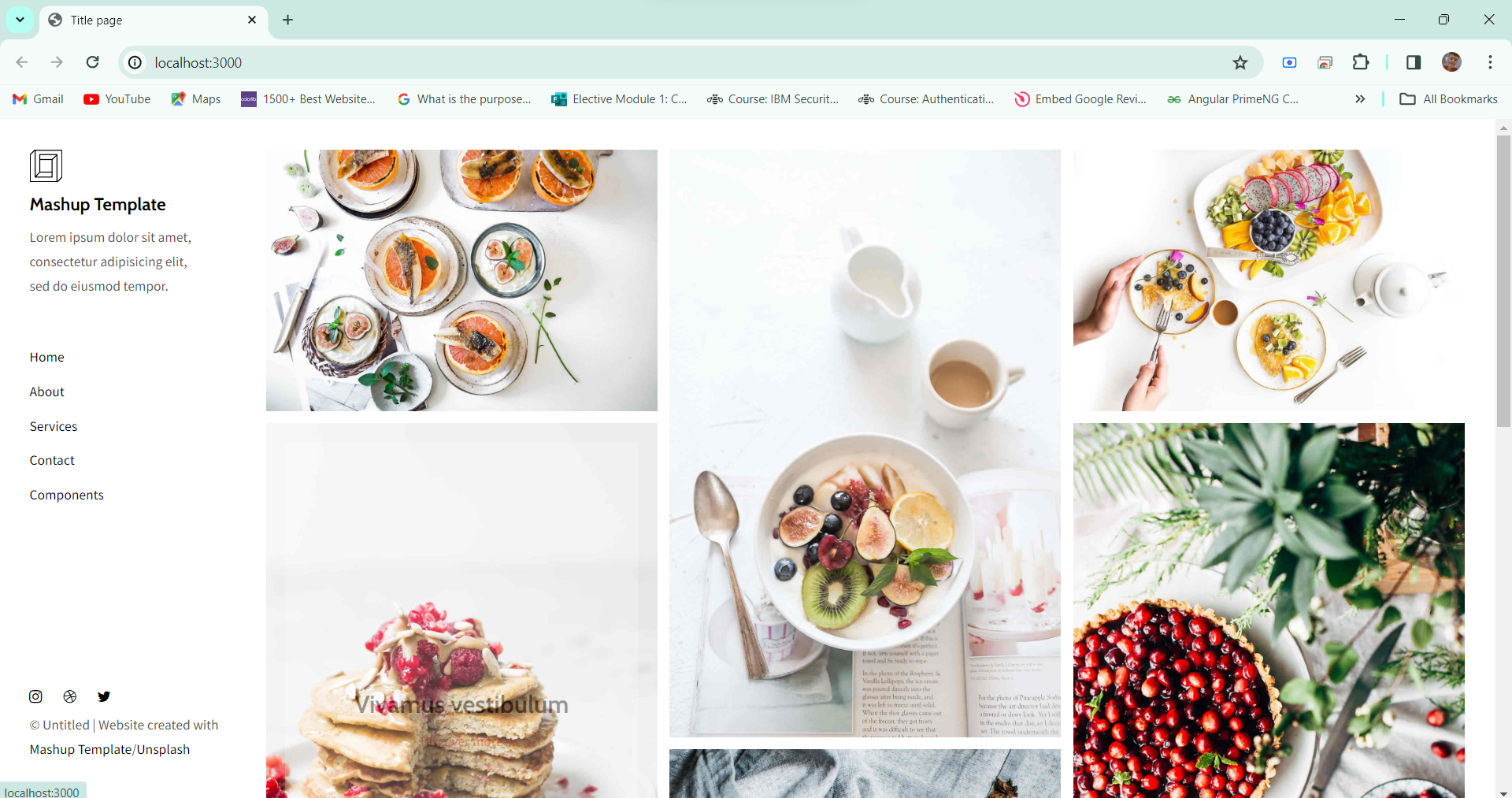
nodemon index.js



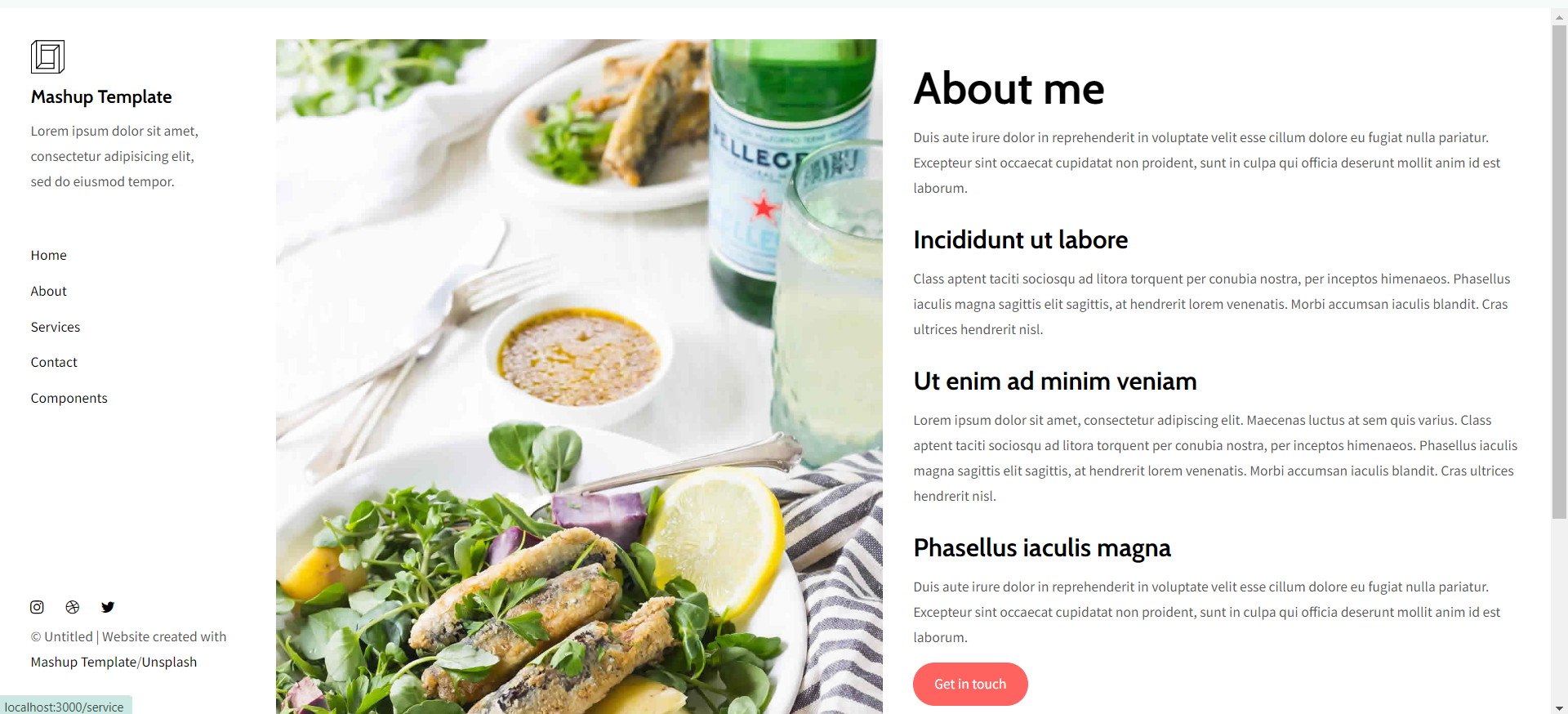
1. Click on the give link.

Output:

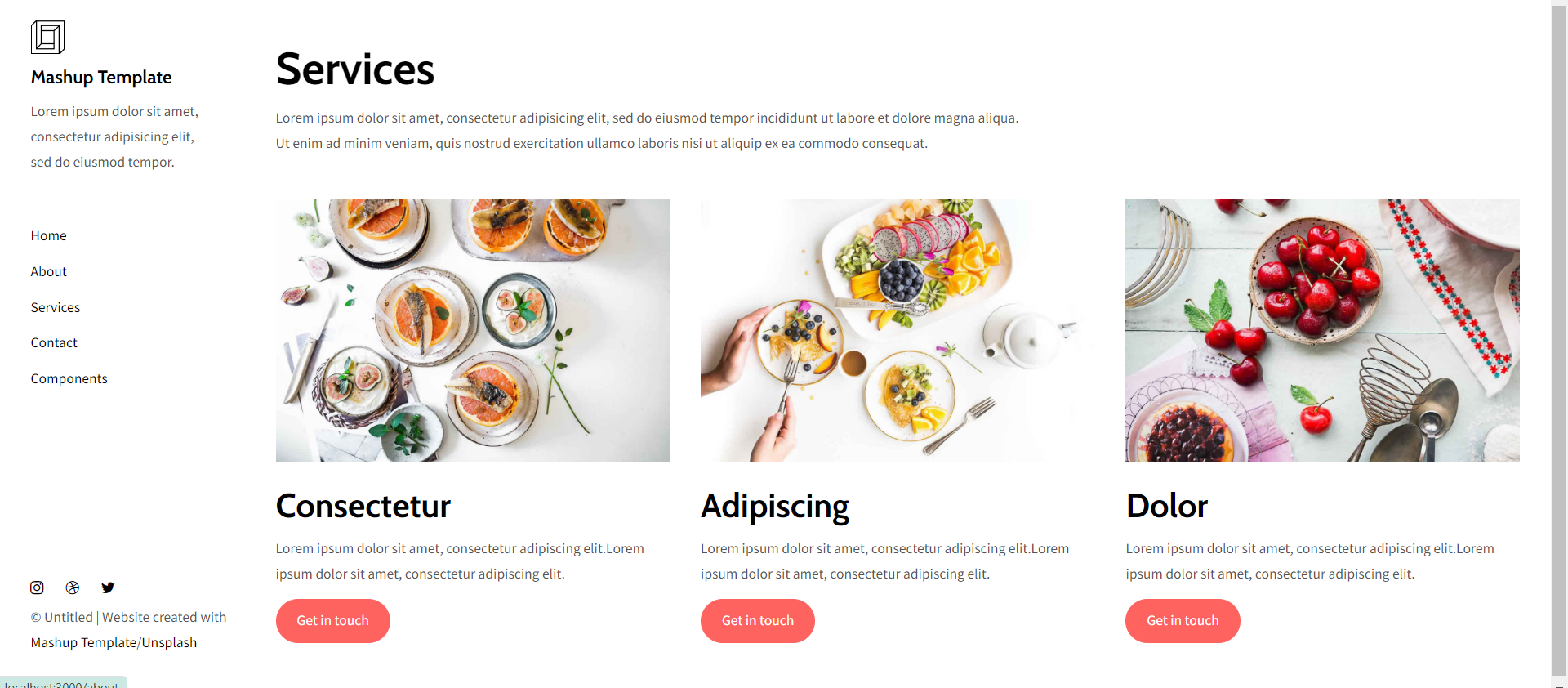
Home Page



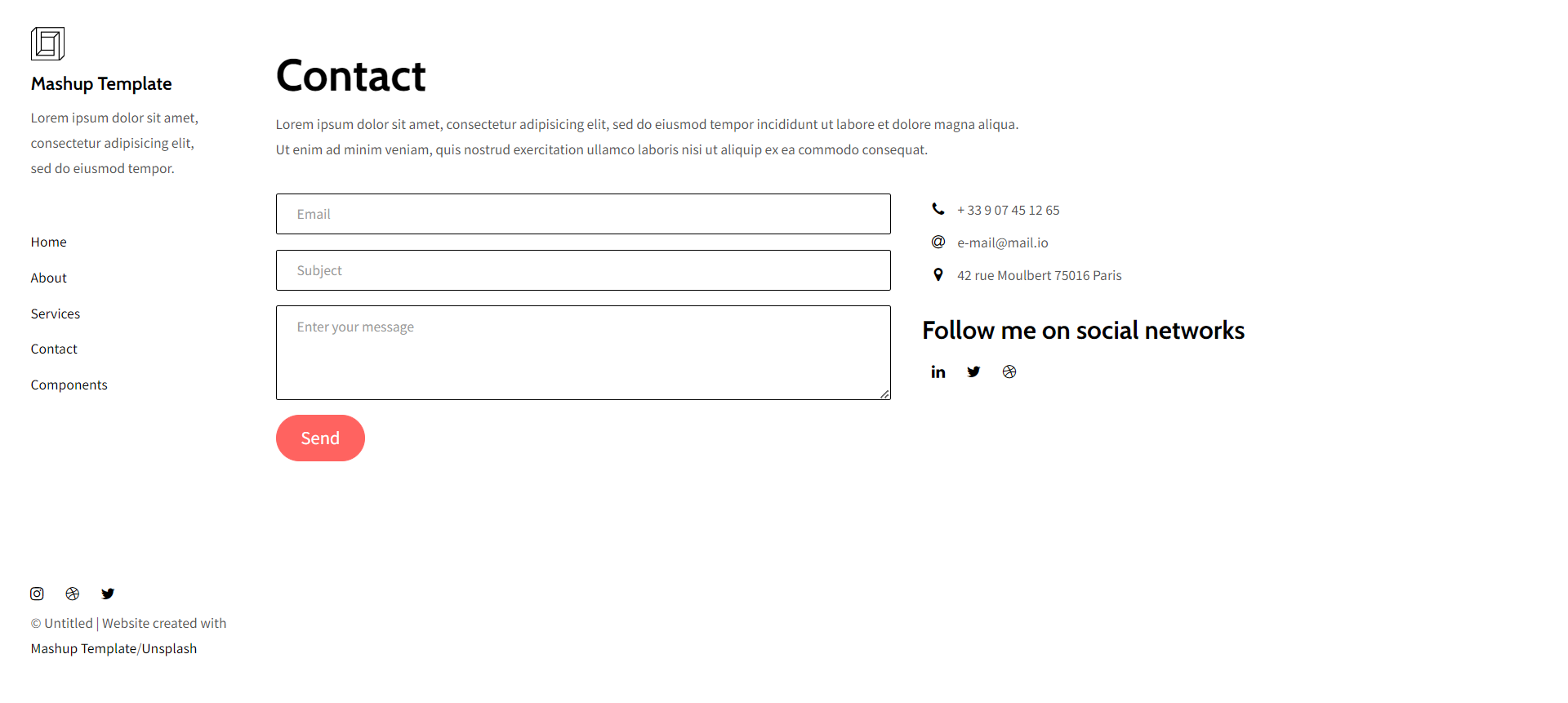
About Page



Services Page



Contact Page



## Activity 3

**Aim:** Create RestAPI using express JS.

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

###### Duration: 4 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS
* ExpressJS

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

1. Before creating RestAPI we required some packages.

**npm init -y**

**npm install express**

**npm install nodemon**

**npm i body-parser**

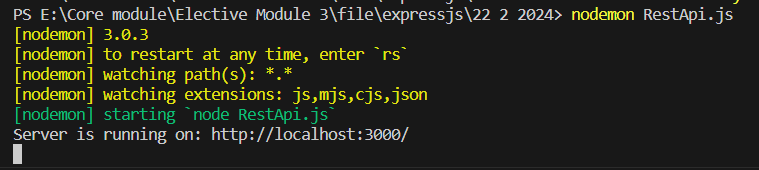
**npm i uuid**

1. Make a file name RestApi.js using Visual Studio Code.

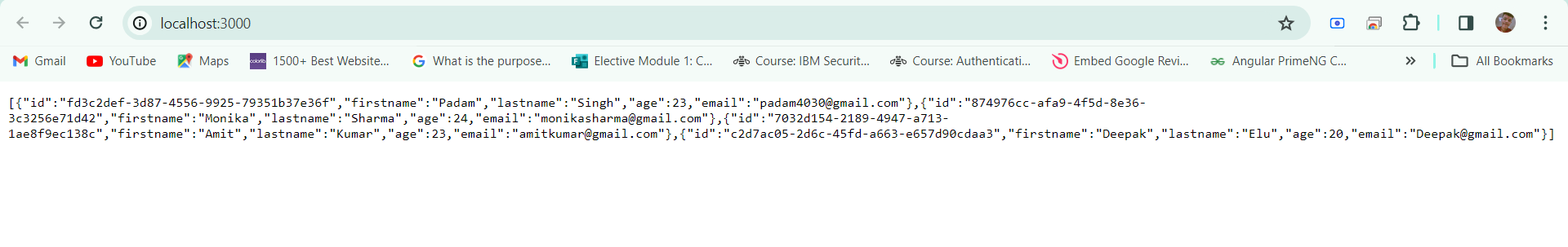
Input:

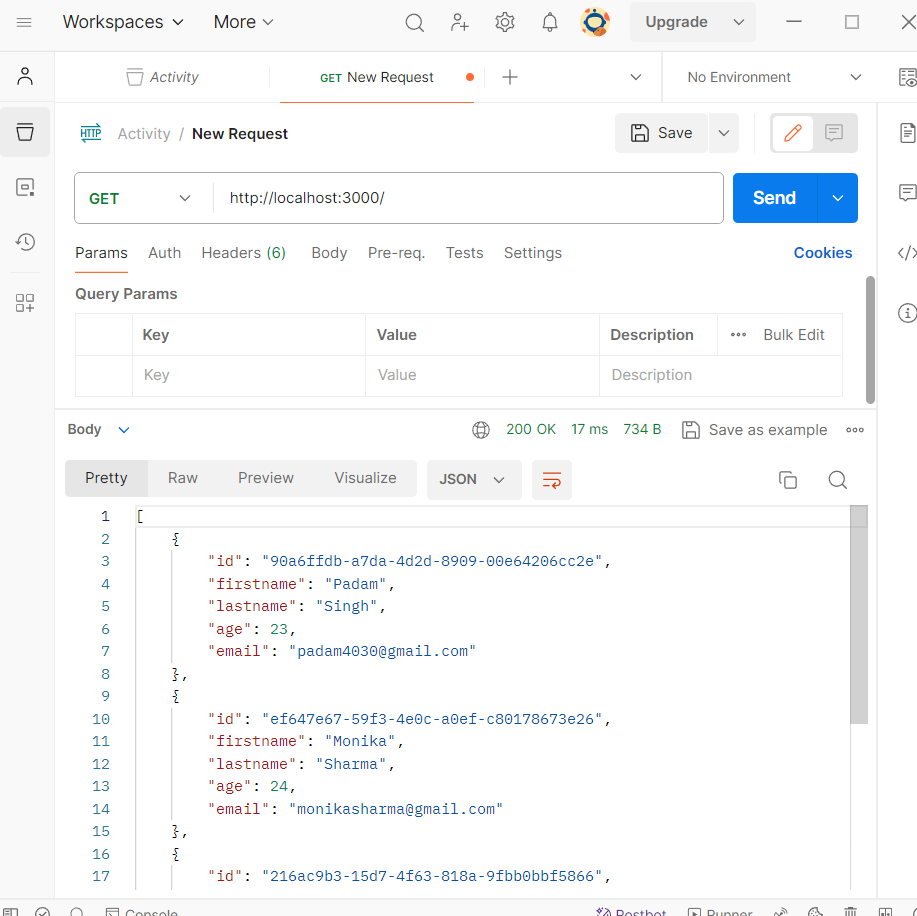
|  |
| --- |
| const express = require('express');  const bodyParser=require("body-parser");  const {v4:uuidv4} = require('uuid');  //create express application instance  const app = express();  port = 3000;  //middleware setup  app.use(bodyParser.json()); //pass the json bodies  //sample data  let users=[  {  id:uuidv4(),  firstname:"Padam",  lastname:"Singh",  age:23,  email:"padam4030@gmail.com",    },  {  id:uuidv4(),  firstname:"Monika",  lastname:"Sharma",  age:24,  email:"monikasharma@gmail.com",    },  {  id:uuidv4(),  firstname:"Amit",  lastname:"Kumar",  age:23,  email:"amitkumar@gmail.com",    },  {  id:uuidv4(),  firstname:"Deepak",  lastname:"Elu",  age:20,  email:"Deepak@gmail.com",    },  ];  //Define routes  //get methond to retrieve all users  app.get('/',(req,res)=>{  res.send(users); //send the array of users as the response  });  app.post("/", (req,res)=>{  //extract data from the request body  const{  firstname,  lastname,  age,  email  }= req.body;  // check if all the required fields are present  if(!firstname ||!lastname ||!age ||!email){  res.status(400).send("Please fill all the fields");  }  else{  //create a new user object with a unique id  const newUser={  id:uuidv4(),  firstname,  lastname,  age,  email  };  //push the new user object to the array of users  users.push(newUser);  //send the newly created user object as the response  res.send(newUser);  }  });  //get method to retrieve a user by id  app.get('/:id',(req,res) => {    //extract the user id from the url parameter  const {id} = req.params;    //find the user in the users array using the provided id and send it back as the response    const foundUser =users.find(user => user.id === id);  //if the user is not found, return error 404  if(!foundUser){  res.status(404).send("User not found");  }  else{  res.send(foundUser);  }  });  //Delete the user from the users array using the provided id  app.delete("/:id", (req,res)=>{  const {id}=req.params;  const removedUser=users.remove(user=>user.id===id);  if(!removedUser){  res.status(404).send();  }else{  res.send(200);  }  });  app.put("/: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 to update a user id  app.patch("/:id",(req,res)=>{  const {id}=req.params;  const {firstname, lastname,age, email}=req.body;  const userIndex = users.findIndex((user)=>user.id===parseInt(id));  //if the user not found throw an error  if(userIndex=== -1){  return res.status(404).json({message: "User not found"});  }  // update the user object with the provided field  if(firstname){  users[userIndex].firstname=firstname;  }  if(lastname){  users[userIndex].lastname=lastname;  }  if(age){  users[userIndex].age=age;  }  if(email){  users[userIndex].email=email;  }  //send a success response with the updated users information  res.json(users[userIndex]);  });  app.listen(port, ()=>{  // Log a message to the console indicating that the server is running  console.log("Server is running on: http://localhost:3000/");  }); |

1. Right click on the folder and open in terminal.
2. Now run the application using nodemon RestApi.js and click on the give link.



Output:





## Learning Outcome 6

## Activity 1

**Aim:** Create Rest api & test rest api using postman.

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

###### Duration: 2 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS
* ExpressJS

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

1. We need to create a folder then install all required packages.

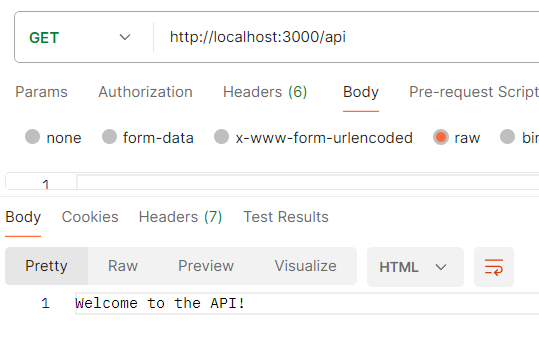
npm init -y

npm i express nodemon

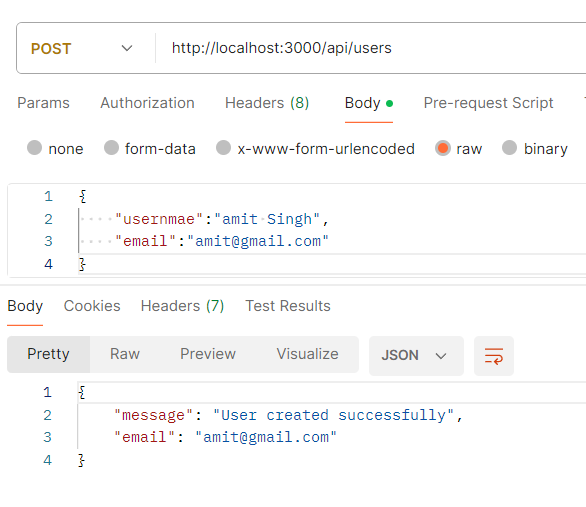
1. After successful installation we need to create a .js file my file name is script.js.
2. Insert these code inside of script.js.

|  |
| --- |
| const express = require('express');  const bodyParser = require('body-parser');  const app = express();  const port = 3000;  app.use(bodyParser.json());  // Define your routes  app.get('/api', (req, res) => {  res.send('Welcome to the API!');  });  app.post('/api/users', (req, res) => {  const { username, email } = req.body;  // Here you can do whatever you want with the received data  res.json({ message: 'User created successfully', username, email });  });  // Start the server  app.listen(port, () => {  console.log(`Server is running on port ${port} http://localhost:${port}`);  }); |

Output:



Create user through postman.



## Learning Outcome 7

## Activity 1

**Aim:** Create application with CRUD operations using Mongo DB.

# **Learning outcome**: Able to configure embedded Mongo DB application with Node JS.

###### Duration: 2 hours

# List of Hardware/Software requirements:

* Operating System – Windows 10/11 or Linux
* Command Prompt/Power Shell
* Text Editor – Notepad / IDE – Visual Studio Code/Any IDE
* NodeJS
* ExpressJS
* MongoDB

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