NAME: Shreyas Parimi
Roll No: Y21CS164

CS325-1(JOEL02)::LBD Course R-20 ::
FULL STACK DEVELOPMENT

Lab 1:

Create a Node.JS environment with node and npm utilities commands and to check and test the node environment with Node.js Console module.

- **Step 1: installation of Node.js environment Node**

1)Download Node.js:

1)Visit the official Node.js website using your web browser.

2)On the homepage, you'll typically find download links for the latest version of Node.js. If you need a specific version, you might need to visit the "Previous releases" section.

3)Choose the appropriate installer for your operating system. Node.js provides installers for various platforms including Windows, macOS, and Linux.

2)Install Node.js:

1)Once the installer is downloaded, locate the downloaded file (usually in your "Downloads" folder).

Follow the installation instructions specific to your operating system:

Windows:

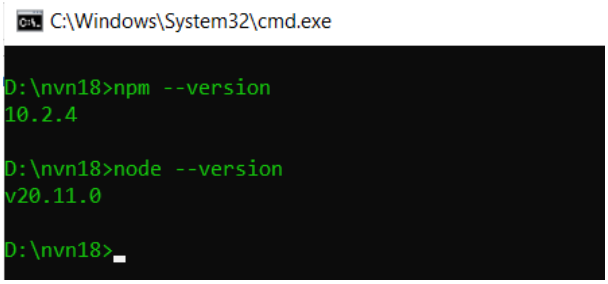
Double-click the downloaded installer file.

Follow the setup wizard instructions. You can generally accept the default settings, but ensure that the "npm package manager" option is selected during installation.

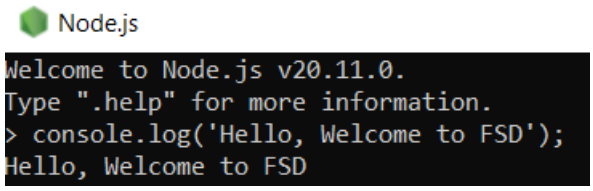
3)Verify Installation:

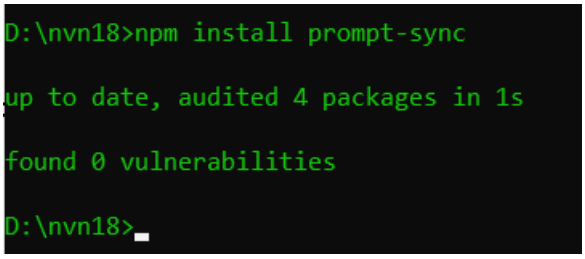
After installation, open a terminal or command prompt.

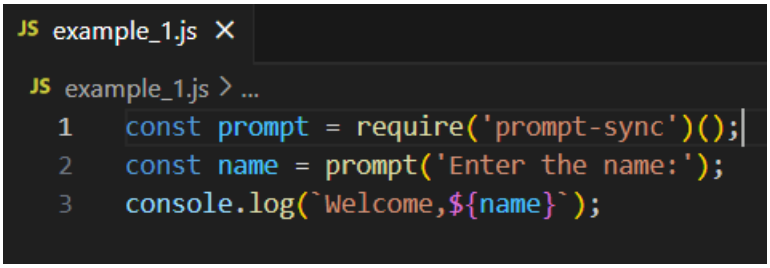
Type the following command to verify that Node.js and npm are installed correctly:

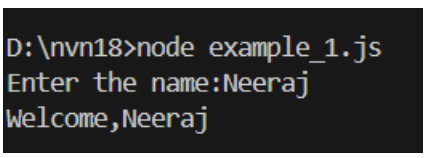


These commands will display the versions of Node.js and npm installed on your system. If you see version numbers for both, the installation was successful.

- **Step 2: Test through the node REPL shell commands**
  - **Open Terminal/Command Prompt:** Open your terminal or command prompt.
  - **Check Node.js Installation**: Type node -v and press Enter to check if Node.js is installed correctly. It should display the version number.
  - **Open Node.js REPL**: Type node and press Enter to open the Node.js REPL (Read-Eval-Print Loop).
  - **Test Node.js Commands**: You can now test JavaScript commands directly in the REPL.
  - 

- **Step-3: install prompt-sync module using npm utility.**
  - Install prompt-sync: In your terminal or command prompt, type:
  - 

- **Step-4: Test and check the prompt-sync with console Module Application**
  - 

  - 

NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 2:

Create a custom Date module using exports keyword Node module by using npm commands and to determine and display current Node.JS Webserver time and date.

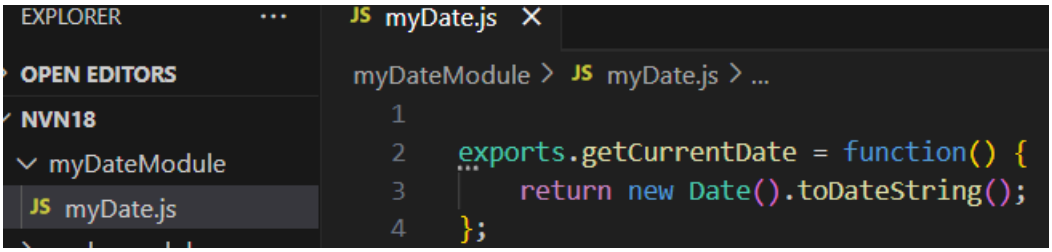
- **Step 1: Create Node Package Module myDate() using node utilities without package.json file**

  1)**Create a Directory**: Create a directory where you want to store your custom Node module. You can name it something like myDateModule.

  2)**Create myDate.js File**: Inside the directory, create a JavaScript file named myDate.js.

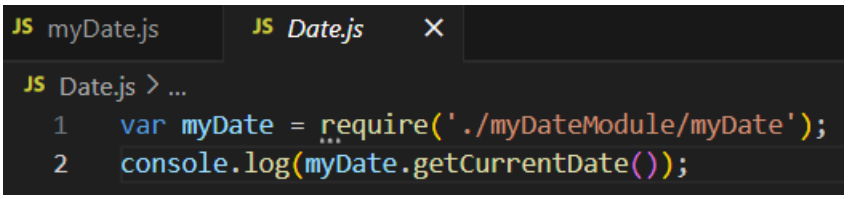
  3)**Define the Module**: In myDate.js, define your custom Date module.

  4)**Export the Module**: Use the exports keyword to make the getCurrentDate function accessible outside the module.




```
D:\nvn18>cd myDateModule

D:\nvn18\myDateModule>node myDate.js

D:\nvn18\myDateModule>cd ..
```

  5)**Using the Module**: Now, you can use this module in other Node.js files by requiring it:




```
D:\nvn18>node Date.js
Thu Apr 11 2024
```

- **Step -2 : Create the Node Package Module myDate() using with package.json file directives like version,name,bin,etc.,**

NAME: Shreyas Parimi
Roll No: Y21CS164

- Initialize a new npm project with npm init This will create ap package.json file.
- In the package.json file, you can specify directives like version, name, bin, etc

```
D:\nvn18\myDateModule>npm init -y
Wrote to D:\nvn18\myDateModule\package.json:

{
  "name": "mydatemodule",
  "version": "1.0.0",
  "description": "",
  "main": "myDate.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

- **Step – 3: Also install created packaged module using npm utility**

  1)To install the module locally, you can use npm install <folder>. For example, if your module is in a folder named mydate, you would use npm install ./myDateModule.

```
D:\nvn18>npm install ./myDateModule
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: 'super@0.1.0',
npm WARN EBADENGINE   required: { node: '>= 0.6.0 < 0.7.0' },
npm WARN EBADENGINE   current: { node: 'v20.11.0', npm: '10.2.4' }
npm WARN EBADENGINE }

added 1 package, and audited 8 packages in 2s

found 0 vulnerabilities
```

  2)To use the module in your code, you can now use require('myDateModule').

Date.js

```
var myDate = require('myDateModule');
console.log(myDate.getCurrentDate());
```

```
D:\nvn18>node Date.js
Thu Apr 11 2024
```

NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 3:

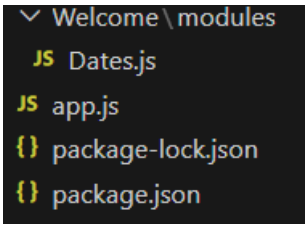
Create Node JS Application with Folder structure using npm utilities and develop one application to display "welcome Node JS APP" Greet message

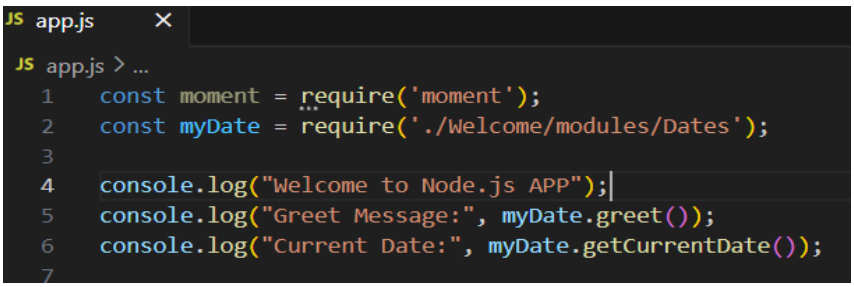
- **Step -1 : With VisualStudioCode APP Framework(Any other)**

1) Create a Folder named Welcome and Create the another Folder inside the Welcome named Modules , inside the Modules create the file named Dates.js in Vs Code.


```
D:\nvn18\Welcome\modules>node Dates.js

D:\nvn18\Welcome\modules>
```

2) Create the app.js file at Outside the Welcome Folder and run the program using the node .






```
D:\nvn18\Welcome\modules>cd ..

D:\nvn18\Welcome>cd ..

D:\nvn18>node app.js
Welcome to Node.js APP
Greet Message: Hello!NVN
Current Date: April 11th 2024, 9:09:07 pm
```

NAME: Shreyas Parimi
Roll No: Y21CS164

- **Step – 2: Without VisualStudioCode APP Framework**

  1) Create a Directory named Welcome and Create the another Directory inside the Welcome named Modules , inside the Modules create the file named Dates.js without Vs Code.

```
D:\nvn18>mkdir Welcome

D:\nvn18>cd Welcome

D:\nvn18\Welcome>mkdir modules

D:\nvn18\Welcome>cd modules

D:\nvn18\Welcome\modules>
```

```
const moment = require('moment');
function greet() {
    return "Hello!NVN, Let's Code";
}
function getCurrentDate() {
    return moment().format('MMMM Do YYYY, h:mm:ss a');
}
module.exports = {
    greet,
    getCurrentDate
};
```

```
D:\nvn18\Welcome\modules>node Dates.js

D:\nvn18\Welcome\modules>
```

  2) Outside the Welcome Directory Create the app.js File to run the main program**:**

```
app.js
app.js > ...
1   const moment = require('moment');
2   const myDate = require('./Welcome/modules/Dates');
3
4   console.log("Welcome to Node.js APP");
5   console.log("Greet Message:", myDate.greet());
6   console.log("Current Date:", myDate.getCurrentDate());
7
```

```
C:\Windows\System32\cmd.exe

D:\nvn18\Welcome\modules>cd ..

D:\nvn18\Welcome>cd ..

D:\nvn18>node app.js
Welcome to Node.js APP
Greet Message: Hello!NVN, Let's Code
Current Date: April 11th 2024, 9:35:07 pm
```

- **Step – 3: Also Access the Custom myDate Module.**

  1) Create a file named myDates.js , where this code exports a function myDates.js that returns current date and time.

NAME: Shreyas Parimi
Roll No: Y21CS164

```
JS myDates.js ×   JS apps.js
JS myDates.js > myDates > myDates
1   exports.myDates = function()
2   {
3       return Date();
4   }
```

```
D:\nvn18>node myDates.js

D:\nvn18>
```

2)import the myDates.js using the require in the apps.js file :

```
JS apps.js > ...
1   var dt = require('./myDates.js');
2
3   console.log('welcome  to NODEJS');
4   console.log('Current time and Date is:'+dt.myDates());
```

```
D:\nvn18>node apps.js
welcome  to NODEJS
Current time and Date is:Thu Apr 11 2024 21:55:11 GMT+0530 (India Standard Time)
```

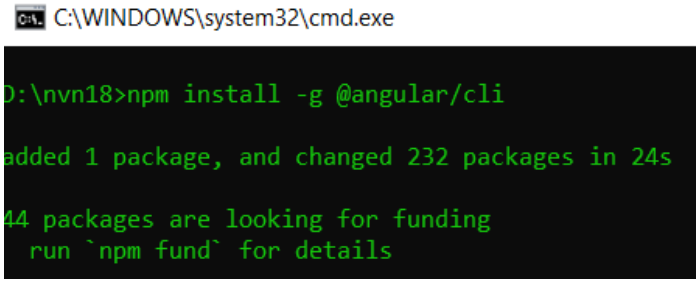
NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 4:

Create Angular CLI Applications with different component configuration steps using different @Angular ng module utilities at CLI environment.

- **Step -1 :Class component Angular app**

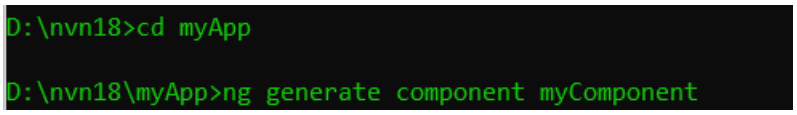
  1) First install the angularjs using the command prompt as show in the below**:**

  2) Create a new Angular Application named myApp with the following syntax**:**

  3) Create a new Component with name of the myComponent:

- **Step – 2: Define Inline selector component in Angular HelloWorld app with root element**

  1) Open my-component.component.ts in your Angular application and by default the selector value will be ‘app-component’ but change the selector property to 'app-root'. Your my-component.component.ts file should look like this:

```
App > src > app > my-component > my-component.component.ts > MyComponentC
1   import { Component } from '@angular/core';
2
3   @Component({
4     selector: 'app-root',
5     standalone: true,
6     imports: [],
7     templateUrl: './my-component.component.html',
8     styleUrl: './my-component.component.css'
9   })
10  export class MyComponentComponent {
11
12  }
13
```

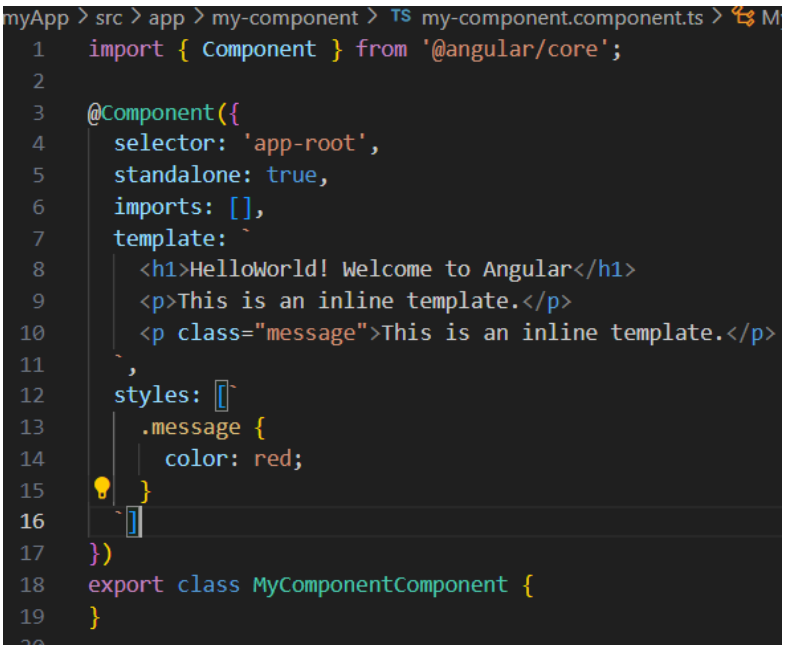
NAME: Shreyas Parimi
Roll No: Y21CS164

- **Step – 3: Define Inline template component in Angular HelloWorld app with HTML elements**

  1) In my-component.component.ts, change the templateUrl property to template and define your HTML elements inline. Your my-component.component.ts file should look like this:

- **Step – 4: Define Inline Style component in Angular HelloWorld app to style the color of the message.**

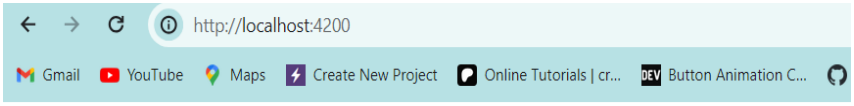
  1) In my-component.component.ts, change the StyleUrl property to styles and define your CSS elements inline. Your my-component.component.ts file should look like this **:**



NAME: Shreyas Parimi
Roll No: Y21CS164

2) Run the application using the following command:

```
D:\nvn18\myApp>ng serve
Initial chunk files | Names         | Raw size
polyfills.js        | polyfills     | 83.60 kB |
main.js             | main          |  1.81 kB |
styles.css          | styles        | 95 bytes |

                    | Initial total | 85.50 kB

Application bundle generation complete. [8.864 seconds]

Watch mode enabled. Watching for file changes...
  ➜  Local:   http://localhost:4200/
  ➜  press h + enter to show help
```



HelloWorld! Welcome to Angular

HTML TEMPLATE - This is an inline template.

STYLE TEMPLATE - This is an inline template.

NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 5:

Create Angular CLI Applications using Angular Class component constructors and objects and different variable initialization.

- **Step – 1: Create Date Class Constructor with current Date in Class Component**

  1)Create a new Angular project: you can create a new Angular project by running:

```
D:\nvn18>ng new Lab --no-standalone
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Static
```

  2)Create a new component: Navigate into your new project directory and generate a new component. For example, if we want to create a component named date, we would run:

```
D:\nvn18>cd Lab

D:\nvn18\Lab>ng g c date
CREATE src/app/date/date.component.html (20 bytes)
CREATE src/app/date/date.component.spec.ts (610 bytes)
CREATE src/app/date/date.component.ts (201 bytes)
CREATE src/app/date/date.component.css (0 bytes)
UPDATE src/app/app.module.ts (543 bytes)
```

  3)Update the Component Class: Open the date.component.ts file and update it as follows:

Lab > src > app > date > TS date.component.ts > ...

```ts
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-date',
  templateUrl: './date.component.html',
  styleUrls: ['./date.component.css']
})
export class DateComponent implements OnInit {
  currentDate: Date;

  constructor() {
    this.currentDate = new Date();
  }

  ngOnInit(): void {
  }
}
```

  After the updation of this file , print the Current Date in the date.component.hmtl file as follows:

NAME: Shreyas Parimi
Roll No: Y21CS164

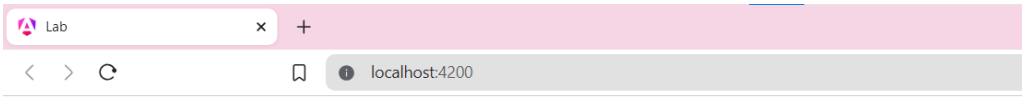
```
Go to component
<h1>Mr.NVN, The date works!</h1>
<h2>the current Date is {{currentDate}}</h2>
```

At this point , the newly generated component date is completed , to run this application , add the <app-date> selector in the app.component.html

```
app.module.ts   app.component.html
Lab > src > app > app.component.html > ...
Go to component
<app-date></app-date>
```

4)Run the Application: To run the application , use the following command:

```
D:\nvn18\Lab>ng serve
Browser bundles
Initial chunk files     | Names          |  Raw size
polyfills.js            | polyfills      |  83.60 kB |
main.js                 | main           |   3.62 kB |
styles.css              | styles         |  95 bytes |

                        | Initial total  |  87.31 kB
```



Mr.NVN, The date works!

the current Date is Fri Apr 12 2024 22:49:19 GMT+0530 (India Standard Time)

- **Step – 2: By using Selector,templateURL and styleURL External component configurations demonstrate the constructor with different objects**

  1)Create a new component: Navigate into your new project directory and generate a new component. Here we created the userDetails component:

```
D:\nvn18\Lab>ng g c users
CREATE src/app/users/users.component.html (21 bytes)
CREATE src/app/users/users.component.spec.ts (617 bytes)
CREATE src/app/users/users.component.ts (205 bytes)
CREATE src/app/users/users.component.css (0 bytes)
UPDATE src/app/app.module.ts (626 bytes)
```

  2)Update the component class: update the users.component.ts file, like this:

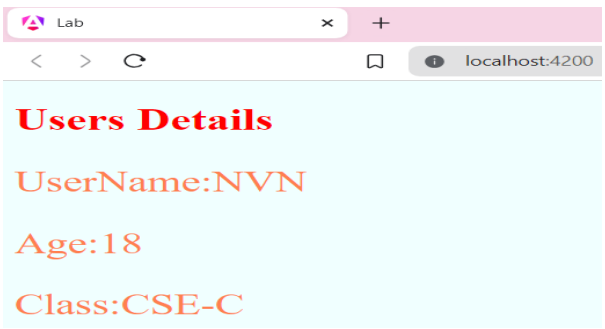
NAME: Shreyas Parimi
Roll No: Y21CS164

```ts
import { Component,OnInit } from '@angular/core';

@Component({
  selector: 'app-users',
  templateUrl: './users.component.html',
  styleUrl: './users.component.css'
})
export class UsersComponent implements OnInit {
  user: {name: string, age: number,class: string};

  constructor() {
    this.user = {
      name: 'NVN',
      age: 18,
      class: 'CSE-C'
    };
  }

  ngOnInit(): void {
  }
}
```

3)Create the html file: update the users.component.html file in the users folder**.**

```html
Go to component
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>Users Details</h1>
    <p>UserName:{{user.name}}</p>
    <p>Age:{{user.age}}</p>
    <p>Class:{{user.class}}</p>
</body>
</html>
```

4)Run the Application: Now, To Run the Application , use the following command:



NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 6:

Create Angular CLI Applications using Angular Expressions and Filters to demonstrate the one App.

- **Create different Angular Expressions in Class Component**
- **Also Specify with Different Angular pipes or filters to demonstrate each filter with Angular expression**

1)Create the angular application with name of Expr_Filter , with the following command :

```
D:\nvn18>ng new Expr_Filter --no-standalone
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and
```

2)Create the new component for the Expr_Filter Application with name of Expression , with the following command :

```
D:\nvn18>cd Expr_Filter

D:\nvn18\Expr_Filter>ng generate component Expression
CREATE src/app/expression/expression.component.html (26 bytes)
CREATE src/app/expression/expression.component.spec.ts (652 bytes)
CREATE src/app/expression/expression.component.ts (225 bytes)
CREATE src/app/expression/expression.component.css (0 bytes)
UPDATE src/app/app.module.ts (567 bytes)
```

3)Write the expression and filters code in the expression.component.ts and bind those values in the express.component.html files , the code follows as:

xpr_Filter > src > app > expression > TS expression.component.ts > ExpressionComponent

```ts
import { Component ,OnInit} from '@angular/core';

@Component({
  selector: 'app-expression',
  templateUrl: './expression.component.html',
  styleUrl: './expression.component.css'
})
export class ExpressionComponent implements OnInit {
  user: {name: string, age: number,class: string};
  title:string;
  currency:number;
  date:Date;
  constructor() {
    this.user = {
      name: 'NVN',
      age: 18,
      class: 'CSE-C'
    };
    this.title='Expression and Filters';
    this.currency = 106106;
    this.date = new Date();
  }
  ngOnInit(): void {
    throw new Error('Method not implemented.');
  }
}
```

NAME: Shreyas Parimi
Roll No: Y21CS164

3)To execute the pipes , write the following code in the expression.component.html in the following way:

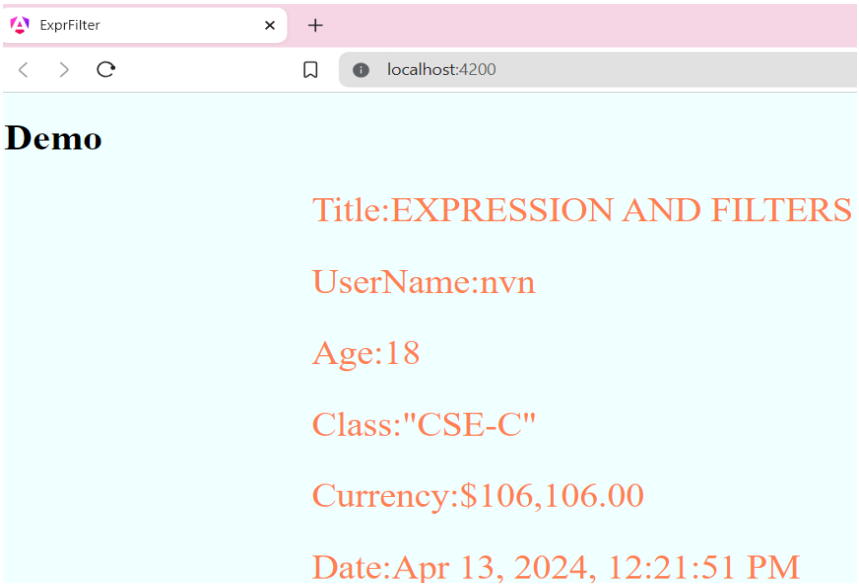
```
Go to component
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <p>Title:{{title|uppercase}}
    <p>UserName:{{user.name | lowercase}}</p>
    <p>Age:{{user.age | number}}</p>
    <p>Class:{{user.class | json}}</p>
    <p>Currency:{{currency|currency}}</p>
    <p>Date:{{date|date:"medium"}}</p>
</body>
</html>
```

4)Before that link the expression.component.html file to app.component.html main file , in the following way:

```
app.component.html    expression.compone
Expr_Filter > src > app > app.component.html > ...
        Go to component
1
2   <app-expression></app-expression>
```

5)Execute the angular application , with the following command :

```
D:\nvn18>cd Expr_Filter

D:\nvn18\Expr_Filter>ng serve
Browser bundles
```



NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 7:

Create Angular CLI Applications using Data Binding demonstrate each binding type with form elements.

- **Interpolation Binding.**
- **Style Binding**
- **Class Binding.**
- **Two –way binding.**

1) **Create the angular application with name binding , in the following way:**

```
D:\nvn18>ng new binding --no-standalone
? Which stylesheet format would you like to use? CSS
? Do you want to enable Server-Side Rendering (SSR) and Stat
```

2) **In the app.component.ts file make the following changes to do the binding process:**

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = `Interpolation||
           Style Binding||
           Class Binding||
           Two-Way Binding`;
  color='blue';
  isSpecial: boolean = true;
  inputValue='';
}
```

3) **All the binding modules will work , except the Two-Way binding , so make the changes in the app.module.ts file.**
In the app.module.ts file , import the FormsModule in that file , in the following way:

```
import { NgModule } from '@angular/core';
import { BrowserModule, provideClientHydration } from '
import { FormsModule } from '@angular/forms';

import { AppRoutingModule } from './app-routing.module'
import { AppComponent } from './app.component';

@NgModule({
  declarations: [
    AppComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule
  ],
  providers: [
    provideClientHydration()
  ],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

NAME: Shreyas Parimi
Roll No: Y21CS164

4) **In the app.component.html , write the following code to binding the elements into webpage:**

app.component.html | app.component.css | app.component.ts | app.module.ts

inding > src > app > app.component.html > html > body

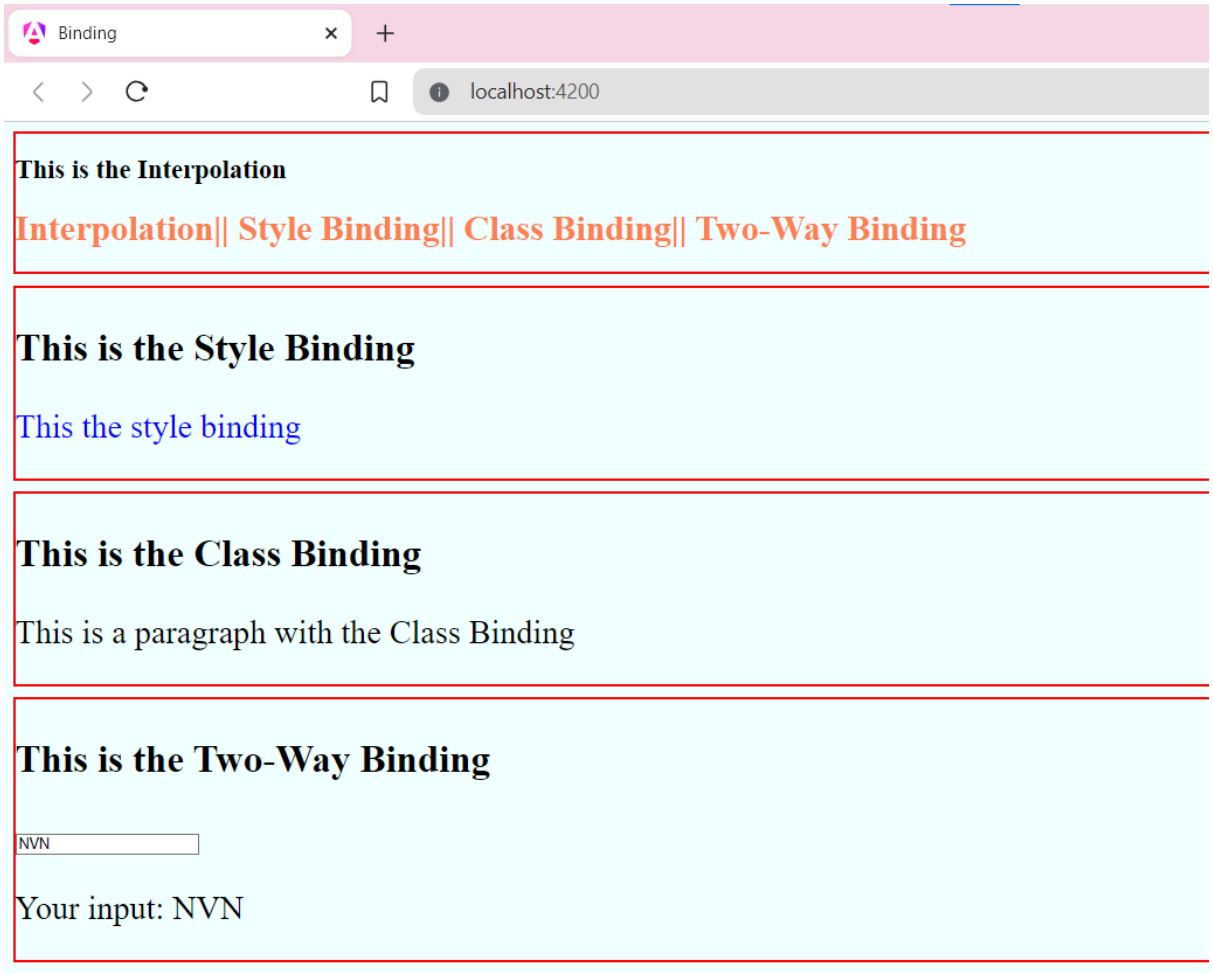
Go to component

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <div id="interpolation">
    <!--Interpolation Binding-->
    <h2>This is the Interpolation</h2>
    <h1>{{title}}</h1>
  </div>

  <div id="style">
    <!--Style Binding-->
    <h3>This is the Style Binding</h3>
    <p [style.color]="color">This the style binding</p>
  </div>

  <div id="class">
    <!--Class Binding-->
    <h3>This is the Class Binding</h3>
    <p [class.special]="isSpecial">This is a paragraph with the Class Binding</p>
  </div>
  <div id="two-way">
    <!--Two-Way Binding-->
    <h3>This is the Two-Way Binding</h3>
    <input [(ngModel)]="inputValue" placeholder="Enter text">
    <p>Your input: {{ inputValue }}</p>
  </div>
</body>
</html>
```

5) **To Run the Application do the following :**

```
D:\nvn18>cd binding

D:\nvn18\binding>ng serve
Browser bundles
Initial chunk files     | Names
polyfills.js            | polyfills
main.js                 | main
```

NAME: Shreyas Parimi
Roll No: Y21CS164



NAME: Shreyas Parimi
Roll No: Y21CS164

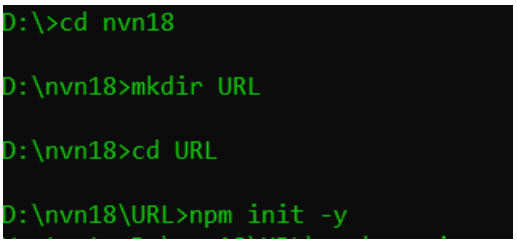
Lab 8:

Create Node.js Application using URL module to decompose URL Components with

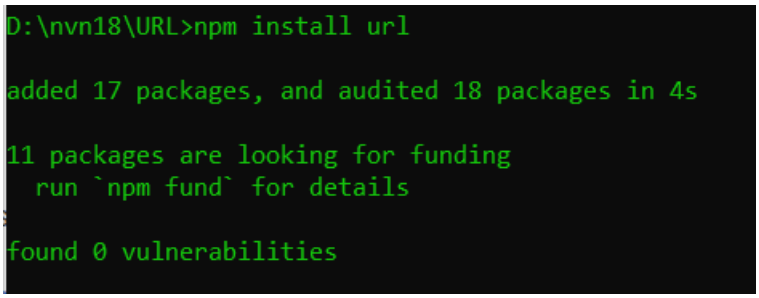
```
urlStr =
'http://user:pass@host.com:80/resource/path?query=string#ha"
```

- Resolving the URL Components with url.parse() and url.format() methods
- Also to Resolving the URL using url.resolve();

1) **Create a New Node.js Project:** Open your terminal and create a new directory for your project. Then navigate into that directory and initialize a new Node.js project by running:



2) **Install Required Dependencies:** Since we'll be using built-in Node.js modules, there are no external dependencies to install.



3) **Create the Node.js Script:** Create a new JavaScript file, named **url.js** such as app.js, in your project directory. This file will contain the code to decompose and resolve URLs.

   Write the code in the following ways:

NAME: Shreyas Parimi
Roll No: Y21CS164

```
const url = require('url');
const urlStr = 'http://user:pass@host.com:80/resource/path?query=string#ha';
const parsedUrl = url.parse(urlStr);

console.log('Decomposed URL Components:');
console.log('Protocol:', parsedUrl.protocol);
console.log('Username:', parsedUrl.auth.split(':')[0]);
console.log('Password:', parsedUrl.auth.split(':')[1]);
console.log('Host:', parsedUrl.host);
console.log('Port:', parsedUrl.port);
console.log('Path:', parsedUrl.pathname);
console.log('Query:', parsedUrl.query);
console.log('Hash:', parsedUrl.hash);
console.log('-------------------------------------------------------');
const resolvedUrl = url.format(parsedUrl);
console.log('Resolved URL:');
console.log(resolvedUrl);
console.log('-------------------------------------------------------');
const resolvedPath = url.resolve('http://example.com/', '/resource');
console.log('Resolved Path:');
console.log(resolvedPath);
```

4) **Run the Application: Run the url.js application , in the following way:**

```
D:\nvn18\URL>node url.js
Decomposed URL Components:
Protocol: http:
Username: user
Password: pass
Host: host.com:80
Port: 80
Path: /resource/path
Query: query=string
Hash: #ha
-------------------------------------------------------
Resolved URL:
http://user:pass@host.com:80/resource/path?query=string#ha
-------------------------------------------------------
Resolved Path:
http://example.com/resource
```

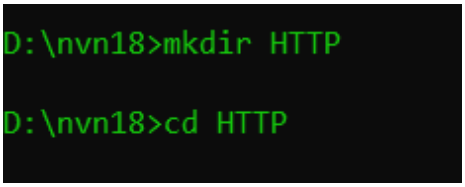
NAME: Shreyas Parimi
Roll No: Y21CS164

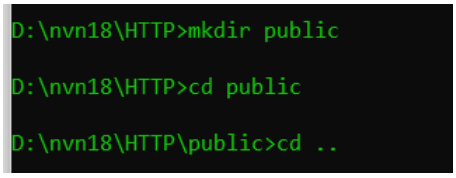
Lab 9:

Implementing Http Server and Http Client using http node.js module and demonstrate the Http Client/server Application.

- Create Http Static server files data using static files.
- Define HttpRequest/HttpResponse objects.

1) **Create the necessary files**: Create a new directory for your project and create the http_server.js and http_client.js files. Also, create a public directory and place server.html file in it. You can put some basic HTML content in server.html.





2) Create the http_server.js files and http_client.js files and write the following code Mentioned below and at the same time create the server.html
3) **Run the server:** Open a terminal, navigate to your project directory, and run node http_server.js. You should see the message 'Server is listening on port 3000'.
4) **Test the server:** Open a web browser and go to http://localhost:3000. You should see the content of your index.html file.
5) **Run the client**: In a new terminal window (or tab), navigate to your project directory and run node http_client.js. You should see the content of your server.html file printed in the terminal. This is the response from the server.

http_server.js:

NAME: Shreyas Parimi
Roll No: Y21CS164

url.js | http_client.js | http_server.js | server.html

HTTP > http_server.js > ...

```
1   const http = require('http');
2   const fs = require('fs');
3   const path = require('path');
4
5   const server = http.createServer((req, res) => {
6       // Serve static files
7       const filePath = path.join(__dirname, 'public', req.url === '/' ? 'server.html' : req.url);
8       const contentType = getContentType(filePath);
9       // Check if file exists
10      fs.readFile(filePath, (err, content) => {
11          if (err) {
12              if (err.code === 'ENOENT') {
13                  res.writeHead(404, { 'Content-Type': 'text/html' });
14                  res.end('<h1>404 Not Found</h1>');
15              } else {
16                  res.writeHead(500);
17                  res.end(`Server Error: ${err.code}`);
18              }
19          } else {
20              res.writeHead(200, { 'Content-Type': contentType });
21              res.end(content, 'utf8');
22          }
23      });
24  });
25  const PORT = process.env.PORT || 3000;
26
27  server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
28  // Function to determine content type based on file extension
29  function getContentType(filePath) {
30      let extname = path.extname(filePath);
31      switch (extname) {
32          case '.html':
33              return 'text/html';
34          case '.js':
35              return 'text/javascript';
36          case '.css':
37              return 'text/css';
38          case '.json':
39              return 'application/json';
40          case '.png':
41              return 'image/png';
42          case '.jpg':
43              return 'image/jpg';
44          default:
45              return 'text/plain';
46      }
47  }
```

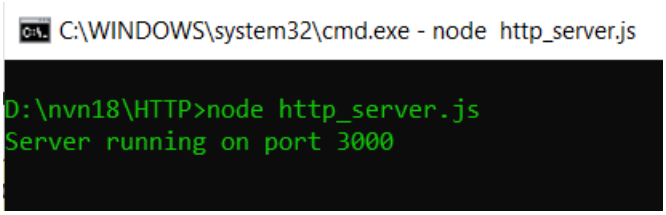
(parameter) err: NodeJS.ErrnoException

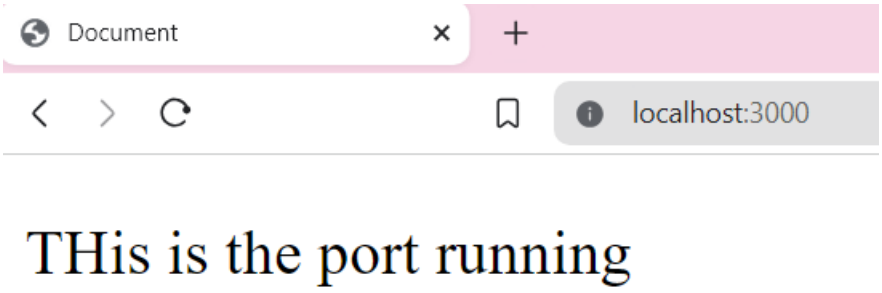
NAME: Shreyas Parimi
Roll No: Y21CS164

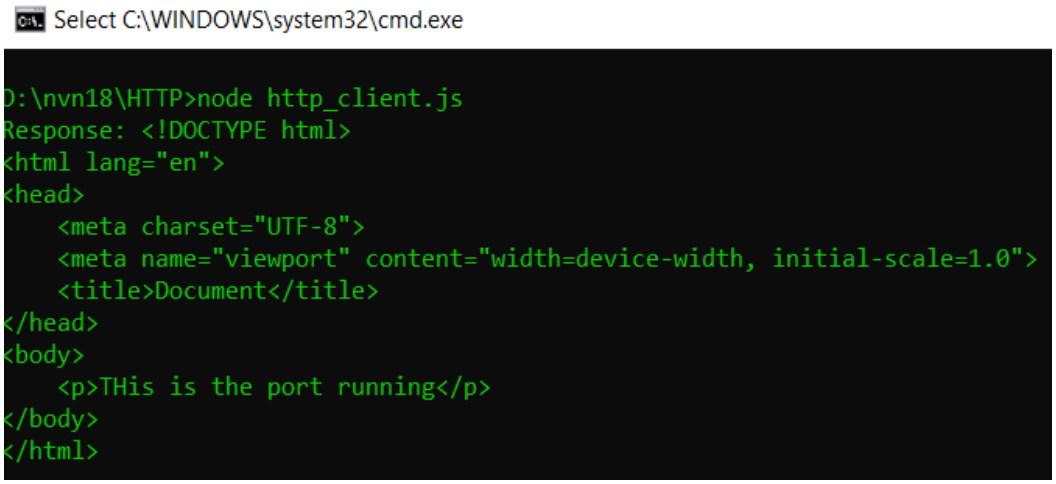
http_client.js:

Server.html:

NAME: Shreyas Parimi
Roll No: Y21CS164







NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 10.

Create Simple Arithmetic Operations Form with different form input elements N1 and N2 text components and ADD button component.

- provide Express Server with listen port:3000
- Use Express.use route and URL Pattern '/add'
- provide different routing configurations either POST or GET

1) Create the directory ADD and inside the directory create the node application with name of add.js .

```
D:\nvn18>mkdir ADD

D:\nvn18>cd ADD
```

```
D:\nvn18>npm install express

added 64 packages, and audited 67 packages in 5s

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

found 0 vulnerabilities
```

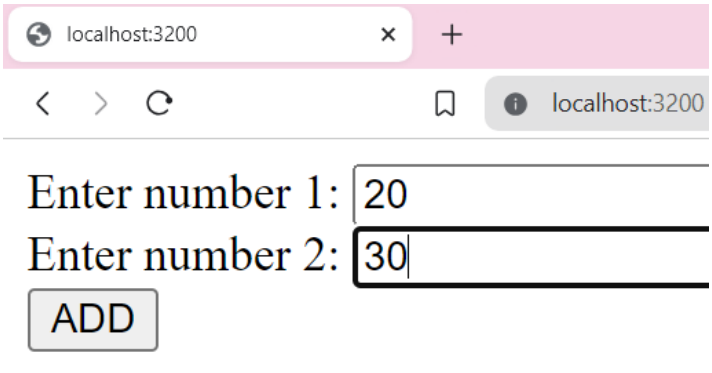
2) We import Express and the body-parser middleware for parsing form data.
3) We create an instance of the Express application.
4) We set the port to 3000.
5) We use bodyParser.urlencoded() middleware to parse URL-encoded form data.
6) We define a route for serving the HTML form (/), which contains two input fields for numbers (N1 and N2) and a submit button (ADD).
7) We define a route for handling POST requests to /add. When the form is submitted, this route extracts the numbers from the form data, adds them together, and sends the result as a response.
8) We optionally define a route for handling GET requests to /add. This route is similar to the POST route but expects the numbers to be passed as query parameters instead of form data.
9) We start the server and listen on port 3000.
10) The code will be followed as :

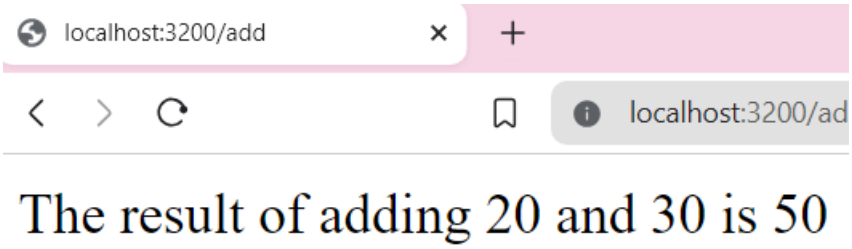
NAME: Shreyas Parimi
Roll No: Y21CS164

ADD > JS add.js > ...

```
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const PORT = 3200;
app.use(bodyParser.urlencoded({ extended: true }));
app.get('/', (req, res) => {
    res.send(`
        <form action="/add" method="POST">
            <label for="n1">Enter number 1:</label>
            <input type="text" id="n1" name="n1"><br>
            <label for="n2">Enter number 2:</label>
            <input type="text" id="n2" name="n2"><br>
            <button type="submit">ADD</button>
        </form>
    `);
});
app.post('/add', (req, res) => {
    const n1 = parseFloat(req.body.n1);
    const n2 = parseFloat(req.body.n2);
    const result = n1 + n2;
    res.send(`The result of adding ${n1} and ${n2} is ${result}`);
});
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

```
D:\nvn18\ADD>node add.js
Server is running on http://localhost:3200
```





The result of adding 20 and 30 is 50

NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 11:

Create Simple Login form Page Application using Express JS Module:

- provide Express Server with listen port:4000 with URL Pattern '/login'
- Display the login form with username, password, and submit button on the screen.
- Users can input the values on the form.
- Validate the username and password entered by the user.
- Display Invalid Login Credentials message when the login fails.
- Show a success message when login is successful.

1) Create the directory named LOGIN and inside the directory create the file named login.js

```
D:\nvn18>mkdir LOGIN

D:\nvn18>cd LOGIN
```

```
D:\nvn18>npm install express

added 64 packages, and audited 67 packages in 5s

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

found 0 vulnerabilities
```

2) We import Express and the body-parser middleware for parsing form data.
3) We create an instance of the Express application.
4) We set the port to 4000.
5) We use bodyParser.urlencoded() middleware to parse URL-encoded form data.
6) We define a route for serving the login form (/login). This route displays a form with input fields for username and password, along with a submit button.
7) We define a route for handling POST requests to /login. This route receives the submitted form data, validates the username and password, and sends an appropriate response:
8) If both username and password are provided and match the expected values (in this case, 'admin' and 'password'), it sends a success message.
9) If either username or password is missing, it sends a message prompting the user to enter both.
10) If the provided username or password is incorrect, it sends an error message indicating invalid credentials.
11) We start the server and listen on port 4000.
12) The code follows as :

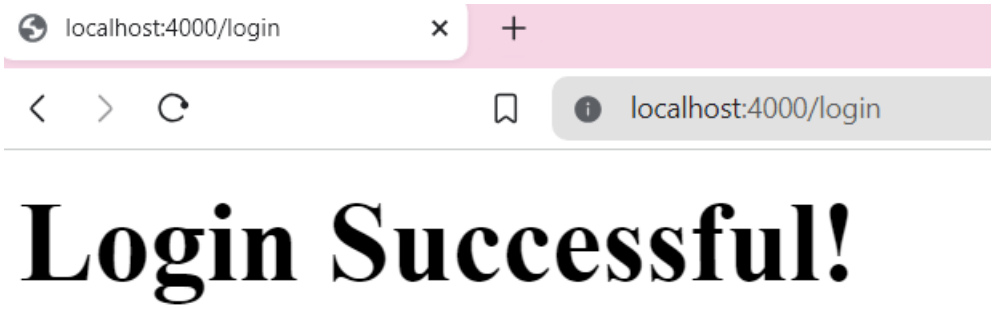
NAME: Shreyas Parimi
Roll No: Y21CS164

Login.js:

```
LOGIN > JS login.js > ...
1   const express = require('express');
2   const bodyParser = require('body-parser');
3   const app = express();
4   const PORT = 4000;
5   app.use(bodyParser.urlencoded({ extended: true }));
6   app.get('/login', (req, res) => {
7       res.send(`
8           <h1>Login</h1>
9           <form action="/login" method="POST">
10              <label for="username">Username:</label>
11              <input type="text" id="username" name="username"><br>
12              <label for="password">Password:</label>
13              <input type="password" id="password" name="password"><br>
14              <button type="submit">Login</button>
15          </form>
16      `);
17  });
18  app.post('/login', (req, res) => {
19      const { username, password } = req.body;
20      if (username && password) {
21          if (username === 'nvn' && password === 'emma') {
22              res.send('<h1>Login Successful!</h1>');
23          } else {
24              res.send('<h1>Invalid Login Credentials</h1>');
25          }
26      } else {
27          res.send('<h1>Please enter both username and password</h1>');
28      }
29  });
30  app.listen(PORT, () => {
31      console.log(`Server is running on http://localhost:${PORT}`);
32  });
```

```
D:\nvn18\LOGIN>node login.js
Server is running on http://localhost:4000
```

NAME: Shreyas Parimi
Roll No: Y21CS164



NAME: Shreyas Parimi
Roll No: Y21CS164

Lab 12:

Create Simple MongDB Server with mongod configuration data and also manage Mongoshell using mongosh :

- **Create simple student document Database**
- **Insert one student record in mongosh**
- **Update and delete one document in mongosh**
- **Also to perform connection from MongoDB to node.js driver connection string**

Step 1: Install MongoDB First, you need to install MongoDB on your machine. You can download it from the official MongoDB website. After downloading, follow the instructions to install it**.**

Step 2: Start MongoDB Server You can start the MongoDB server by running the mongod command in your terminal. This will start the MongoDB server on the default port 27017.

```
C:\Users\DELL>mongod
{"t":{"$date":"2024-04-16T19:06:30.084+05:30"},"s":"I",
d wire specification","attr":{"spec":{"incomingExternalCl
lClient":{"minWireVersion":0,"maxWireVersion":21},"outgoi
:true}}}
{"t":{"$date":"2024-04-16T19:06:30.088+05:30"},"s":"I",
```

Step-3: Connect to MongoDB Server using Mongoshell Open a new terminal window and connect to the MongoDB server using the mongo command. This will start the MongoDB shell (mongosh).

```
C:\Users\DELL>mongosh
Current Mongosh Log ID: 661e7f4e6dcde486d7117b7a
Connecting to:          mongodb://127.0.0.1:27017/?directConnectio
.2.4
Using MongoDB:          7.0.8
Using Mongosh:          2.2.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-04-16T18:47:26.981+05:30: Access control is not enabled fo
iguration is unrestricted
------

test>
```

Step – 4: Create the DataBase named the Student in the mongosh shell using the following command . Before creating the database first establish the connection with the MongoDB compass and place the localhost address in the mongoshell .

NAME: Shreyas Parimi
Roll No: Y21CS164

```
Please enter a MongoDB connection string (Default: mongodb://localhost/): mongodb://localhost:27017
mongodb://localhost:27017
Current Mongosh Log ID: 661e7bd148573ec585117b7a
Connecting to:          mongodb://localhost:27017/?directConnection=true&serverSelectionTimeoutMS=2000&
.2.4
Using MongoDB:          7.0.8
Using Mongosh:          2.2.4

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

------
   The server generated these startup warnings when booting
   2024-04-16T18:47:26.981+05:30: Access control is not enabled for the database. Read and write access
iguration is unrestricted
------
```

```
neeraj> use Student
switched to db Student
Student>
```

Step – 5: Insert the values into the documents under the database named Student:

```
Student> db.details.insertOne({name:"neeraj",age:"18"})
{
  acknowledged: true,
  insertedId: ObjectId('661e838748573ec585117b7c')
}
```

```
Student> db.details.insertMany([
... {
...  name:"Tayyab",
... age:19
... },
... {
...  name:"Roshan",
... age:19
... },
... {
...  name:"Sathish",
... age:19
... }
... ]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('661e850048573ec585117b80'),
    '1': ObjectId('661e850048573ec585117b81'),
    '2': ObjectId('661e850048573ec585117b82')
  }
}
```

NAME: Shreyas Parimi
Roll No: Y21CS164

Step-6 : Update the values in the documents named details under the database Student.

```
Student> db.details.updateOne({name:"neeraj"},{$set:{age:20}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Step- 7 : Delete the any values in the details in the Database named Student:

```
Student> db.details.deleteOne({name:"neeraj"})
{ acknowledged: true, deletedCount: 1 }
Student>
```

Step – 8: Connecct the MongoDB server to Nodejs Application , first you need to install the mongodb in the nodejs using the npm

```
D:\nvn18>npm install mongodb

added 12 packages, and audited 82 packages in 13s

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

found 0 vulnerabilities
```

```
D:\nvn18>npm  install mongoose

added 8 packages, and audited 90 packages in 6s

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

found 0 vulnerabilities
```

you can use the MongoClient object to connect to your MongoDB server:

1) Create the mongoo.js application and write the following code
2) Make sure that the mongoserver is running and make the connections
3) Print the all the values in the database.

NAME: Shreyas Parimi
Roll No: Y21CS164

mongoo.js > ...

```js
const mongoose = require('mongoose');
const uri = 'mongodb://localhost:27017/Student';
mongoose.connect(uri, { useNewUrlParser: true, useUnifiedTopology: true });
const db = mongoose.connection;
db.on('error', console.error.bind(console, 'MongoDB connection error:'));
db.once('open', async () => {
  console.log('Connected to MongoDB');
  const detailsSchema = new mongoose.Schema({
    name: String,
    age: Number,
  });
  const Details = mongoose.model('Details', detailsSchema);
  try {
    const docs = await Details.find({});
    console.log(docs);
  } catch (error) {
    console.error('Error fetching documents:', error);
  } finally {
    mongoose.connection.close();
  }
});

```

```
D:\nvn18>node mongoo.js
Connected to MongoDB
[
  {
    _id: new ObjectId('661e850048573ec585117b80'),
    name: 'Tayyab',
    age: 19
  },
  {
    _id: new ObjectId('661e850048573ec585117b81'),
    name: 'Roshan',
    age: 19
  },
  {
    _id: new ObjectId('661e850048573ec585117b82'),
    name: 'Sathish',
    age: 19
  },
  {
    _id: new ObjectId('661e855d48573ec585117b83'),
    name: 'Neeraj',
    age: 18
  }
]
```