NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

CS325(JOEL02)::R-20::FULLSTACKDEVELOPMENT

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab1:

CreateaNode.JSenvironmentwithnodeandnpmutilitiescommandsandtocheckand test the node environment with Node.js Console module.

- **Step1:installationofNode.jsenvironmentNode**

1) DownloadNode.js:

1) VisittheofficialNode.jswebsiteusingyour webbrowser.
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) InstallNode.js:

1) Oncetheinstaller isdownloaded, locatethedownloadedfile(usually inyour "Downloads" folder).

Followtheinstallationinstructionsspecifictoyouroperatingsystem:

Windows:

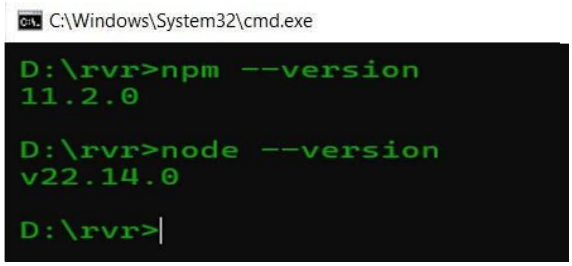
Double-clickthedownloadedinstaller file.

Follow the setup wizard instructions. You can generally accept the defaultsettings, butensurethatthe"npmpackagemanager"optionis selected during installation.

3) VerifyInstallation:

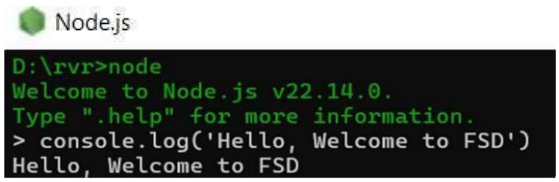
Afterinstallation,openaterminalorcommandprompt.

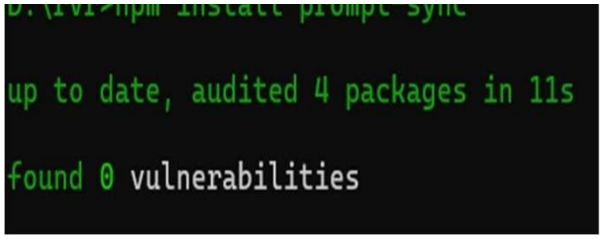
Typethefollowingcommandtoverifythat Node.js and npm areinstalled correctly:



ThesecommandswilldisplaytheversionsofNode.jsandnpminstalledonyour system. If you see version numbers for both, the installation was successful.

- **Step2:TestthroughthenodeREPLshellcommands**
  - **OpenTerminal/CommandPrompt:**Openyourterminalorcommand prompt.
  - **CheckNode.jsInstallation**:Typenode-vandpress Enter tocheckifNode.js is installed correctly. It should display the version number.
  - **OpenNode.js REPL**:Typenodeandpress Enter toopentheNode.js REPL (Read-Eval-Print Loop).
  - **TestNode.jsCommands**:YoucannowtestJavaScriptcommandsdirectlyin the REPL.



- **Step-3:installprompt-syncmoduleusingnpmutility.**
  - Installprompt-sync:Inyourterminalor commandprompt,type:
  - 

- **Step-4:TestandcheckthepromptsyncwithconsoleModuleApplication**

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab2:

Create a custom Date module using exports keyword Node module by using npm commands andtodetermineanddisplaycurrentNode.JSWebservertimeanddate.

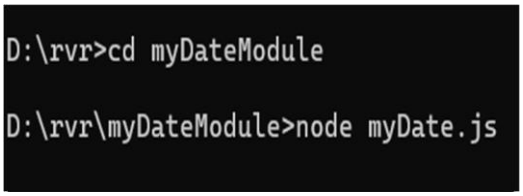
- **Step1:CreateNodePackageModule myDate()usingnodeutilitieswithout package.json file**

1) **Create aDirectory**:Createa directorywhereyou want tostoreyour custom Node module. You can name it something like myDate Module.

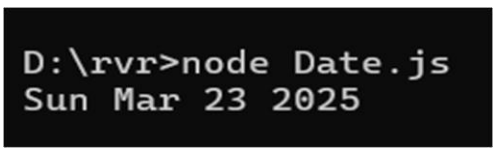
2) **Create myDate.jsFile**:Insidethedirectory,createa JavaScriptfilenamed myDate.js.

3) **DefinetheModule**:InmyDate.js,defineyourcustomDatemodule.

4) **ExporttheModule**:Usetheexports keywordtomaketheget Current Date function accessible outside the module.



5) **UsingtheModule**:Now,youcanusethis moduleinother Node.jsfilesby requiring it:



- **Step-2:CreatetheNodePackageModule myDate()usingwithpackage.jsonfile directives like version, name,bin,etc.,**
  - Initializea newnpmproject withnpminitThiswillcreatea package.json file.
  - Inthepackage.jsonfile,youcanspecifydirectiveslikeversion,name,bin,etc

```
{
  "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:Alsoinstallcreatedpackagedmoduleusingnpmutility**

  1) To install the module locally, you can use npm install <folder>. For example,ifyour moduleisinafoldernamedmydate,youwouldusenpm install ./myDateModule.

```
D:\rvr>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) Tousethemoduleinyour code, youcannowuserequire('myDateModule').

Date.js

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

```
D:\rvr>node Date.js
Sun Mar 23 2025
```

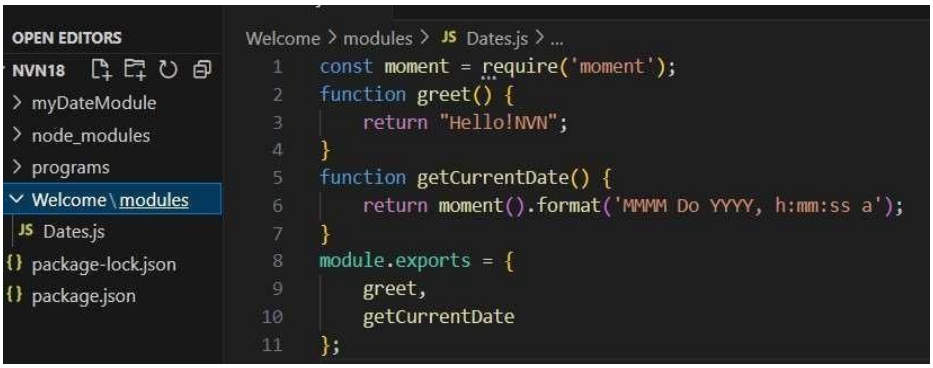
NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

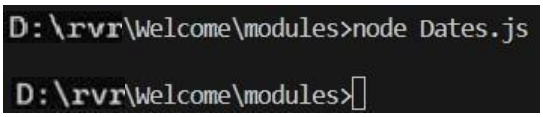
Lab3:

CreateNodeJSApplicationwithFolderstructureusing npmutilities anddevelopone application to display “welcome Node JS APP” Greet message

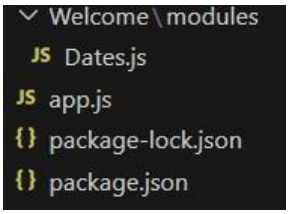
- **Step-1:WithVisualStudioCodeAPPFramework(Anyother)**

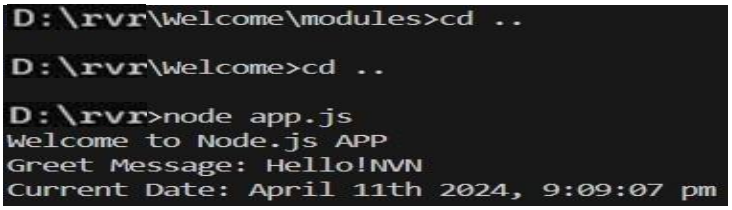
1) Create a Folder named Welcome and Create the another Folder inside the WelcomenamedModules, insidetheModulescreatethefilenamedDates.js in Vs Code.





2) Createtheapp.js fileat OutsidetheWelcomeFolder andrun theprogram using the node .





- **Step–2:WithoutVisualStudioCodeAPPFramework**

  1) Createa DirectorynamedWelcomeandCreatetheanother Directoryinsidethe Welcome named Modules , inside the Modules create the file named Dates.js without Vs Code.

```
D:\rvr>mkdir Welcome

D:\rvr>cd Welcome

D:\rvr\Welcome>mkdir modules

D:\rvr\Welcome>cd modules

D:\rvr\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:\rvr\Welcome\modules>node Dates.js

D:\rvr\Welcome\modules>
```

  2) OutsidetheWelcome DirectoryCreatetheapp.jsFiletorunthemainprogram**:**

```
JS app.js
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:\rvr\Welcome\modules>cd ..

D:\rvr\Welcome>cd ..

D:\rvr>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:AlsoAccesstheCustommyDateModule.**

  1) Createa filenamedmyDates.js,wherethiscodeexportsa functionmyDates.js that returns current date and time.

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

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

D:\rvr>
```

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

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab4:

CreateAngularCLIApplicationswithdifferentcomponentconfigurationsteps using different @Angular ng module utilities at CLI environment.

- **Step-1:ClasscomponentAngularapp**

  1) Firstinstalltheangularjsusingthecommandpromptasshowinthebelow**:**

  ```
  C:\WINDOWS\system32\cmd.exe
  D:\rvr>npm install -g @angular/cli

  added 273 packages in 2m

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

  2) Createa newAngular ApplicationnamedmyApp withthefollowing syntax**:**

  ```
  D:\rvr> ng new myApp
  ```

  3) CreateanewComponentwithnameofthe myComponent:

  ```
  D:\rvr> cd myApp

  D:\rvr\myApp>ng generate component myComponent
  ```

- **Step – 2: Define Inline selector component in Angular HelloWorldapp 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
```

- **Step–3:DefineInlinetemplatecomponentinAngularHelloWorldappwith HTML elements**

  1) Inmy-component.component.ts, changethetemplateUrlpropertytotemplate and define your HTML elements inline. Your my-component.component.ts file should look like this:

myApp > src > app > my-component > TS my-component.component.ts > MyComponentComponent

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

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [],
  template: `
    <h1>HelloWorld! Welcome to Angular</h1>
    <p>This is an inline template.</p>
  `,
  styleUrl: './my-component.component.css'
})
export class MyComponentComponent {

}
```

- **Step–4:DefineInlineStylecomponent inAngular HelloWorldapptostyle the color of the message.**

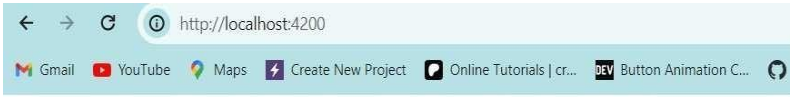
  1) Inmy-component.component.ts, changetheStyleUrlpropertytostylesand define your CSS elements inline. Your my-component.component.ts file should look like this **:**

myApp > src > app > my-component > TS my-component.component.ts > M

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

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [],
  template: `
    <h1>HelloWorld! Welcome to Angular</h1>
    <p>This is an inline template.</p>
    <p class="message">This is an inline template.</p>
  `,
  styles: [`
    .message {
      color: red;
    }
  `]
})
export class MyComponentComponent {
}
```

2) Runtheapplicationusingthefollowingcommand:

```
D:\rvr\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: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab5:

CreateAngularCLIApplications usingAngularClasscomponentconstructorsand objects and different variable initialization.

- **Step–1:CreateDateClassConstructorwithcurrentDateinClassComponent**

1) Create anewAngularproject: youcancreatea newAngular projectby running:

```
D:\rvr>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:\rvr>cd Lab

D:\rvr\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) UpdatetheComponentClass:Openthedate.component.tsfileandupdate 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 theupdationofthisfile,printtheCurrentDateinthe date.component.hmtl file as follows:

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

Atthispoint, thenewlygeneratedcomponentdateiscompleted, torun 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) **RuntheApplication:** Toruntheapplication,usethefollowingcommand:

```
D:\rvr\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
```

- **Step–2:ByusingSelector,templateURLandstyleURL Externalcomponent configurations demonstrate the constructor with different objects**

1) **Create anewcomponent:**Navigateintoyour newproject directoryand generate a new component. Here we created the userDetails component:

```
D:\rvr\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) **Updatethecomponentclass:updatetheusers.component.tsfile,like this:**

```
ab > src > app > users > TS users.component.ts > UsersComponent > cons
1   import { Component,OnInit } from '@angular/core';
2
3   @Component({
4     selector: 'app-users',
5     templateUrl: './users.component.html',
6     styleUrl: './users.component.css'
7   })
8   export class UsersComponent implements OnInit {
9     user: {name: string, age: number,class: string};
10
11    constructor() {
12      this.user = {
13        name: 'NVN',
14        age: 18,
15        class: 'CSE-C'
16      };
17    }
18
19    ngOnInit(): void {
20    }
21  }
```

3) **Createthe htmlfile:**updatetheusers.component.htmlfileintheusers folder**.**

```
ab > src > app > users > users.component.html > html > body > p
    Go to component
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h1>Users Details</h1>
10      <p>UserName:{{user.name}}</p>
11      <p>Age:{{user.age}}</p>
12      <p>Class:{{user.class}}</p>
13  </body>
14  </html>
15
```

4) **RuntheApplication:** Now, ToRuntheApplication,usethefollowing command:

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab6:

CreateAngularCLIApplications usingAngular ExpressionsandFilterstodemonstratethe one App.

- **CreatedifferentAngularExpressionsinClassComponent**
- **AlsoSpecifywithDifferentAngularpipesorfilterstodemonstrateeachfilter with Angular expression**

1) Createtheangular applicationwithnameof Expr_Filter, withthefollowing command :

```
D:\rvr>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) Createthenewcomponentfor theExpr_Filter Applicationwithname of Expression , with the following command :

```
D:\rvr>cd Expr_Filter

D:\rvr\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) Writetheexpressionandfilters codeintheexpression.component.tsand bindthosevaluesintheexpress.component.htmlfiles,thecodefollowsas:

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.');
  }
}
```

3) Toexecutethepipes , writethefollowingcodeinthe 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) Beforethatlinktheexpression.component.htmlfiletoapp.component.html main file , in the following way:

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

5) Executetheangularapplication, withthefollowingcommand:

```
D:\rvr>cd Expr_Filter

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

ExprFilter

localhost:4200

Demo

Title:EXPRESSION AND FILTERS

UserName: Ram

Age:18

Class:"CSE A "

Currency:$106,106.00

Date:Mar 13, 2025, 12:21:51 PM

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab7:

CreateAngularCLIApplications usingDataBindingdemonstrateeachbindingtype with form elements.

- **InterpolationBinding.**
- **StyleBinding**
- **ClassBinding.**
- **Two–waybinding.**

1) **Createtheangularapplicationwithnamebinding,inthe followingway:**

```
D:\rvr>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) **Intheapp.component.tsfilemakethefollowingchangestodothebinding 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) **Allthebinding moduleswillwork,exceptttheTwo-Waybinding,so make the changes in the app.module.ts file.**
Intheapp.module.tsfile, importtheForms Module inthat file, inthe 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: TULAM SAI SUDHEER
RollNo: Y22CS184

4) **Intheapp.component.html,writethefollowingcodetobindingthe elements into webpage:**

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

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

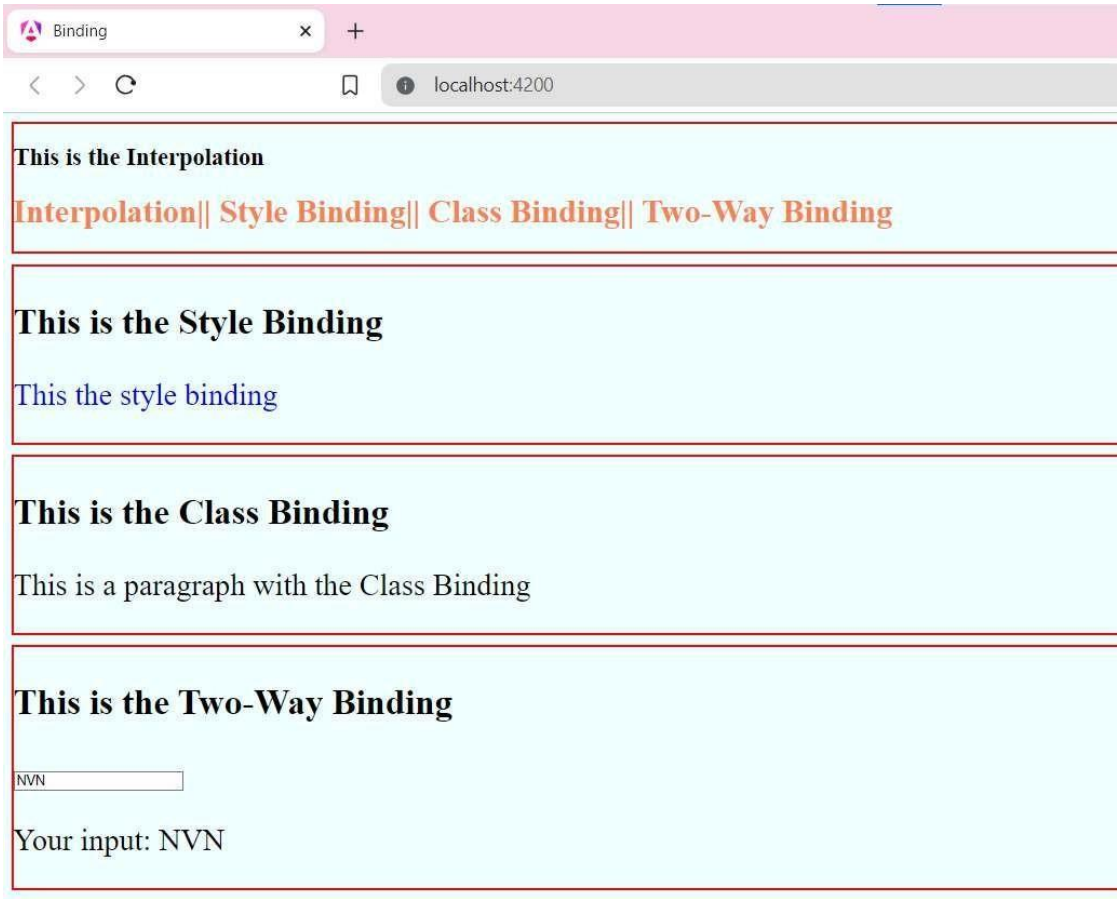
Go to component

```
<!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) **ToRuntheApplicationdothefollowing:**

```
D:\rvr>cd binding

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

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184



NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab8:

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

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

- ResolvingtheURLComponentswithurl.parse()andurl.format()methods
- AlsotoResolvingtheURL usingurl.resolve();

1) **Create aNewNode.jsProject:**Openyour terminalandcreatea newdirectory for your project. Then navigate into that directory and initialize a new Node.js project by running:

```
D:\>cd nvn18

D:\rvr>mkdir URL

D:\rvr>cd URL

D:\rvr\URL>npm init -y
```

2) **InstallRequiredDependencies:**Sincewe'llbeusingbuilt-inNode.js modules, there are no external dependencies to install.

```
D:\rvr\URL>npm install url

added 17 packages, and audited 18 packages in 4s

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

found 0 vulnerabilities
```

3) **Create the Node.js Script:** Create a new JavaScript file, named **url.js** such as app.js,inyour project directory. Thisfilewill containthecodetodecomposeand resolve URLs.

   Writethecodeinthefollowingways:

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

```
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) RuntheApplication:Runthe url.jsapplication, inthefollowingway:

```
D:\rvr\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: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab9:

ImplementingHttpServer andHttpClientusinghttpnode.js moduleanddemonstratethe Http Client/server Application.

- CreateHttpStaticserver filesdatausingstaticfiles.
- DefineHttpRequest/HttpResponseobjects.
    1) **Createthe necessaryfiles**:Createa newdirectoryfor your projectandcreate 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) Cre[illegible]ent.jsfilesandwritethefollowingcode Mentioned below and at the same time create the server.html
3) **Runtheserver:**Opena terminal,navigatetoyour projectdirectory,andrun node http_server.js. You should see the message ‘Server is listening on port 3000’.
4) **Testtheserver:**Opena webbrowser andgotohttp://localhost:3000. Youshould see the content of your index.html file.
5) **Runthe client**:Ina newterminal window(ortab), navigateto your project directory and run node http_client.js. You should see the content of your server.htmlfileprintedintheterminal. This is theresponsefromtheserver.

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

http_server.js:


```
const http = require('http');
const fs = require('fs');
const path = require('path');

const server = http.createServer((req, res) => {
    // Serve static files
    const filePath = path.join(__dirname, 'public', req.url === '/' ? 'server.html' : req.url);
    const contentType = getContentType(filePath);
    // Check if file exists
    fs.readFile(filePath, (err, content) => {
        if (err) {
            if (err.code === 'ENOENT') {
                res.writeHead(404, { 'Content-Type': 'text/html' });
                res.end('<h1>404 Not Found</h1>');
            } else {
                res.writeHead(500);
                res.end(`Server Error: ${err.code}`);
            }
        } else {
            res.writeHead(200, { 'Content-Type': contentType });
            res.end(content, 'utf8');
        }
    });
});
const PORT = process.env.PORT || 3000;

server.listen(PORT, () => console.log(`Server running on port ${PORT}`));
// Function to determine content type based on file extension
function getContentType(filePath) {
    let extname = path.extname(filePath);
    switch (extname) {
        case '.html':
            return 'text/html';
        case '.js':
            return 'text/javascript';
        case '.css':
            return 'text/css';
        case '.json':
            return 'application/json';
        case '.png':
            return 'image/png';
        case '.jpg':
            return 'image/jpg';
        default:
            return 'text/plain';
    }
}
```

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

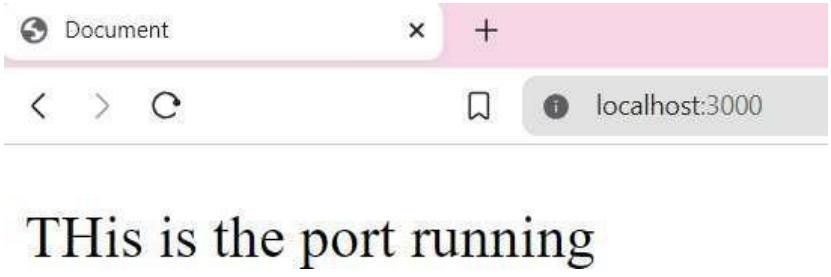
http_client.js:

Server.html:

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

C:\WINDOWS\system32\cmd.exe - node http_server.js

```
D:\rvr\HTTP>node http_server.js
server running on port 3000
```



Select C:\WINDOWS\system32\cmd.exe

```
D:\rvr\HTTP>node http_client.js
Response: <!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>THis is the port running</p>
</body>
</html>
```

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab10.

CreateSimpleArithmeticOperations Formwithdifferentforminput elementsN1andN2 text components and ADD button component.

- ProvideExpressServerwithlistenport:3000
- UseExpress.userouteand URLPattern'/add'
- ProvidedifferentroutingconfigurationseitherPOST orGET

1) CreatethedirectoryADDandinsidethedirectorycreatethenodeapplication with name of add.js .

```
D:\rvr>mkdir ADD

D:\rvr>cd ADD
```

```
D:\rvr>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
```

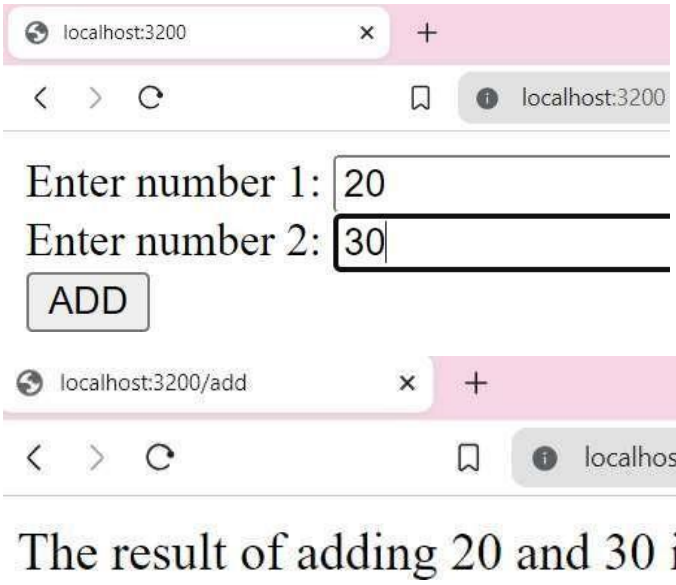
2) WeimportExpress andthebody-parsermiddlewareforparsingformdata.
3) Wecreateaninstance oftheExpressapplication.
4) Wesettheportto 3000.
5) WeusebodyParser.urlencoded()middlewaretoparseURL-encodedformdata.
6) Wedefinea routefor servingtheHTMLform(/), whichcontainstwoinputfields for numbers (N1 and N2) and a submit button (ADD).
7) Wedefinea routefor handlingPOSTrequeststo/add. Whentheformis submitted, this route extracts the numbers from the form data, adds them together, and sends the result as a response.
8) Weoptionallydefinea routefor handlingGETrequeststo/add.Thisrouteis similar to the POST route but expects the numbers to be passed as query parameters instead of form data.
9) Westarttheserverandlistenonport3000.
10) Thecodewillbefollowedas:

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

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



NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab11:

CreateSimpleLoginformPageApplicationusingExpressJSModule:

- ProvideExpressServerwithlistenport:4000with URLPattern‘/login’
- Displaytheloginformwithusername,password,andsubmit buttononthescreen.
- Userscaninput thevaluesontheform.
- Validatetheusernameandpasswordenteredbytheuser.
- DisplayInvalidLoginCredentialsmessagewhentheloginfails.
- Showasuccessmessagewhenloginis successful.

1) CreatethedirectorynamedLOGIN andinsidethedirectory createthefilenamed login.js

```
D:\rvr>mkdir LOGIN

D:\rvr>cd LOGIN
```

```
D:\rvr>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) WeimportExpress andthebody-parsermiddlewareforparsingformdata.
3) Wecreateaninstance oftheExpressapplication.
4) Wesettheportto 4000.
5) WeusebodyParser.urlencoded()middlewaretoparseURL-encodedformdata.
6) Wedefinea routefor servingtheloginform(/login). This routedisplaysa form with input fields for username and password, along with a submit button.
7) Wedefinea routefor handlingPOSTrequests to/login.Thisroutereceivesthe submitted form data, validates the username and password, and sends an appropriate response:
8) Ifbothusernameandpasswordareprovidedandmatchtheexpectedvalues(in this case, 'admin' and 'password'), it sends a success message.
9) Ifeitherusernameorpasswordis missing,itsendsamessagepromptingtheuser to enter both.
10) Iftheprovidedusernameor passwordisincorrect,itsendsanerror message indicating invalid credentials.
11) Westarttheserverand listen onport4000.
12) Thecodefollowsas:

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Login.js:

```
OGIN > JS login.js > ...
const express = require('express');
const bodyParser = require('body-parser');
const app = express();
const PORT = 4000;
app.use(bodyParser.urlencoded({ extended: true }));
app.get('/login', (req, res) => {
    res.send(`
        <h1>Login</h1>
        <form action="/login" method="POST">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username"><br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"><br>
            <button type="submit">Login</button>
        </form>
    `);
});
app.post('/login', (req, res) => {
    const { username, password } = req.body;
    if (username && password) {
        if (username === 'nvn' && password === 'emma') {
            res.send('<h1>Login Successful!</h1>');
        } else {
            res.send('<h1>Invalid Login Credentials</h1>');
        }
    } else {
        res.send('<h1>Please enter both username and password</h1>');
    }
});
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```

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

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

Lab12:

CreateSimpleMongDBServerwithmongodconfigurationdataandalso manage Mongoshell using mongosh :

- **CreatesimplestudentdocumentDatabase**
- **Insertonestudentrecordinmongosh**
- **Updateanddeleteonedocumentinmongosh**
- **AlsotoperformconnectionfromMongoDBtonode.jsdriverconnectionstring**

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 byrunning 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 shell and place the localhost address in the mongoshell .

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

```
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:Insertthevaluesintothedocumentsunder thedatabasenamedStudent:

```
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')
  }
}
```

Step-6: Updatethevaluesinthedocumentsnameddetailsunder thedatabase Student.

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

Step-7: Deletetheanyvalues inthedetailsintheDatabasenamedStudent:

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

Step–8: ConnecttheMongoDBserver toNodejsApplication,firstyou needto 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
```

YoucanusetheMongoClientobjecttoconnecttoyourMongoDBserver:

1) Createthemongoo.jsapplicationandwritethefollowingcode
2) Makesurethatthemongoserverisrunningandmaketheconnections
3) Printallthevaluesinthedatabase.

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184

```
const { MongoClient } = require("mongodb");
const url = "mongodb://localhost:27017";
const client = new MongoClient(url);
async function main() {
    try {
        await client.connect();
        console.log("Connected");
    } catch (err) {
        console.log(err);
    } finally {
        await client.close();
    }
}
main();
```

OUTPUT:
C:\Users\rvr
>nodeapp.js
Connected

InsertthedatausingMongoClientobject:

```
const { MongoClient } = require("mongodb");
async function main() {
    const url = 'mongodb://localhost:27017';
    const client = new MongoClient(url);
    try {
        await client.connect();
console.log("Connected to MongoDB");
  const database = client.db("RVRDB");
      const collection = database.collection("Student");
        const document = [
{ name: "John Doe", age: 22, Ename: "ENG2021001" },
{ name: "Alice Smith", age: 21, Ename: "ENG2021002" },
{ name: "Michael Brown", age: 23, Ename: "ENG2021003" }];
            const result = await collection.insertMany(document);
console.log("Inserted documents:", result.insertedCount);

    } catch (err) {
console.error("Error:", err);
    } finally {
        await client.close();
    }
}
```

OUTPUT:

**C:\Users\rvr>nodea
pp.js Connected to
MongoDB Inserted
documents: 3**

NAME: TULAM SAI SUDHEER
RollNo: Y22CS184