Lab Program 1:

Aim:

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

- Steps for installation of Node.js environment Node
- Test through the node REPL shell commands
- Also install prompt-sync module using npm utility.
- Test and check the prompt-sync with console Module Application

Program:

Steps for installation of Node.js environment Node:

1. Download the Node.js Installer: Visit the official Node.js website - https://nodejs.org/download/ and download `.msi` installer.

2. Run the Node.js Installer: Open the downloaded `.msi` file. If the system prompts for, Do you want to allow this app to make changes to your device? `, click `Yes`. The Node.js Setup wizard will open. Follow the on-screen instructions.

3. Verify the Installation: After the installation, you can verify that Node.js was properly installed by opening your command prompt or Windows PowerShell and running the following command: `node -v`.

Test through the node REPL shell commands:

```
Command Prompt - node
C:\Users\Reguvel>node -v
v18.17.1

C:\Users\Reguvel>npm -v
9.6.7

C:\Users\Reguvel>node
Welcome to Node.js v18.17.1.
Type ".help" for more information.
> .editor
// Entering editor mode (Ctrl+D to finish, Ctrl+C to cancel)
console.log(" This is your Node.js Environment, Welcome!! ");
 This is your Node.js Environment, Welcome!!
undefined
>
```

Install the prompt-sync module:

Open your terminal and navigate to your project directory. Then, run the following command to install the prompt-sync module:

➢ npm install prompt-sync

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

Create a JavaScript file: Create a new JavaScript file in your project directory (for example, name.js). In this file, you can require the prompt-sync module and use it to get input from the user.

name.js:

```
var prompt = require('prompt-sync')( );
var name = prompt('What is your name? ');
console.log('Hi ',name);
```

Run the JavaScript file: In your terminal, run the JavaScript file using Node.js:

➢ node name.js

Output:

```
Command Prompt
C:\Users\Reguvel\FSD Lab\Lab 1>node name.js
What is your name? Reguvel
Hi  Reguvel

C:\Users\Reguvel\FSD Lab\Lab 1>
```

Lab Program 2:

Aim:

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.

• Create Node Package Module myDate() using node utilities without package.json file.

• Also Create the Node Package Module myDate() using with package.json file directives like version, name, bin, etc.,

• Also install created packaged module using npm utility.

Program:

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

1.Create a JavaScript file:

Create a new JavaScript file in your project directory (for example, myDate.js). In this file, you can export a function that returns the current date and time.

myDate.js:

```
exports.myDate = function() {
 return new Date();
};
```

2.Test the myDate module:

Create another JavaScript file (for example, date.js). In this file, you can require the myDate module and use it to print the current date and time.

date.js:

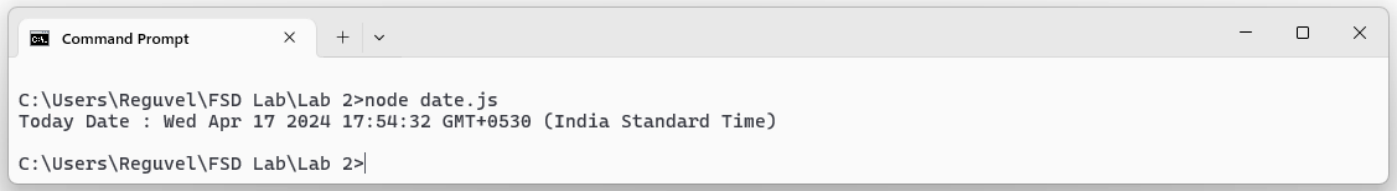
```
var myDate = require('./myDate');
console.log("Today Date : "+myDate.myDate());
```

3.Run the JavaScript file:

In your terminal, run the JavaScript file using Node.js

➢ node date.js

Output:



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

1.Create a JavaScript file:

Create a new JavaScript file in your project directory (for example, myDate.js). In this file, you can export a function that returns the current date and time.

myDate.js:

```
exports.myDate = function(){
 return new Date();
};
```

2.Test the myDate module:

Create another JavaScript file (for example, date.js). In this file, you can require the myDate module and use it to print the current date and time.

date.js:

```
var myDate = require('./myDate');
console.log("Today Date : "+myDate.myDate());
```

3.Create a JSON file:

Create a new JSON file in your project directory (package.json). In this file, you can add name, version, author, description etc.. of your module.

package.json:

```
{
```

```
"name": "myDateModule",
"version": "1.0.0",
"description": "A simple module that returns the current date and time",
"main": "myDate.js",
"keywords": [
"date",
"time"
],
"author": "Reguvel",
}
```

4. Create the Module file:

Navigate to the ../myDate folder in console and run the following command to build a local package module.

Output:

```
Command Prompt

C:\Users\Reguvel\FSD Lab\Lab 2\myDate>npm pack
npm notice
npm notice package: myDateModule@1.0.0
npm notice === Tarball Contents ===
npm notice 56B  myDate.js
npm notice 205B package.json
npm notice === Tarball Details ===
npm notice name:          myDateModule
npm notice version:       1.0.0
npm notice filename:      myDateModule-1.0.0.tgz
npm notice package size:  305 B
npm notice unpacked size: 261 B
npm notice shasum:        301fbffdc92ba9e8586e45ade8b766b8e95cd55d
npm notice integrity:     sha512-IJpCzhcl7wlAy[...]qTEZbzjGgnHOw==
npm notice total files:   2
npm notice
myDateModule-1.0.0.tgz
```

Also install created packaged module using npm utility:

1.Install the module:

From the project directory in console, using the following command install the myDate Module.

- ➢ npm install ../myDate/myDateModule-1.0.0.tgz

Output:

```
Command Prompt

C:\Users\Reguvel\FSDLab\Lab2>npm install C:\Users\Reguvel\FSDLab\Lab2\myDate\myDateModule-1.0.0.tgz

up to date in 458ms

C:\Users\Reguvel\FSDLab\Lab2>node date.js
Today Date : Wed Apr 17 2024 18:34:43 GMT+0530 (India Standard Time)
```

Lab Program 3:

Aim:

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

- With VisualStudioCode APP Framework(Any other).
- Without VisualStudioCode APP Framework.
- Also Access the Custom myDate Module.

Program:

1. Create a new directory:

Create a new directory for your project using following command in your terminal

➢ mkdir myNodeApp

2.Navigate into that directory:

Navigate into your project directory using command

➢ cd myNodeApp

3. Create a new folder named modules:

Inside the modules folder, create a new file named Datetime.js using following commands

➢ mkdir modules

➢ cd modules && notepad Datetime.js

Datetime.js:

```
exports.Datetime = function () {
 return Date();
};
```

4. Navigate back to root directory using following command:

➢ cd..

5. Create a new file named app.js

Create a new js file in your root directory using command

➢ notepad app.js

app.js:

```
var http = require('http');
var date = require('./modules/DateTime');
http.createServer(function (req, res) {
 res.writeHead(200, {'Content-Type': 'text/html'});
 res.write("<h1 style='color:blue;'>Welcome Node JS APP</h1>");
 res.write("<h1 style='color:black;'>Current date and time are: <span
style='color:red;'>" + date.Datetime() + "</span></h1>");
res.end();
}).listen(8080);
```

6. Run the application:

Run the Node JS Application using the following command

➢ node app.js

Output:

Lab Program 4:

Aim:

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

- Class component Angular app
- Define Inline selector component in Angular HelloWorld app with root element
- Define Inline template component in Angular HelloWorld app with HTML elements
- Define Inline Style component in Angular HelloWorld app to style the color of the message.

Program:

1. Create a new Angular project with Angular CLI:

```
ng new ang-app
select CSS and <enter>, enter "y" for yes
```

2. Navigate into the project directory:

```
cd ang-app
```

3. Go to comp/src/app

4. Edit app.component.ts as follows:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
@Component({
 selector: 'app-root',
 standalone: true,
 imports: [RouterOutlet],
 template: `
 <pre>
 <h1 >Hello world</h1>
 <h2 >Created app component</h2>
 <h3 >Inline Template and selector</h3></pre>`,
styles:[`h1{color:blue;} h2{color:red;} h3{color:green; }`]
})
export class AppComponent {
 title = 'comp';
}
```

4. Now run the component:

```
➢ npm run ng serve
```

5. Click on the localhost link:

Output:

Lab Program 5:

Aim:

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

• Create Date Class Constructor with current Date in Class Component

• By using Selector, templateURL and styleURL External component configurations demonstrate the constructor with different objects

Steps:

1. Open comp/src/app
2. Edit the **app.component.html** as follows:

```
<form>
<p>Click this button for the current date:</p>
 <button type="button" (click)="today()">Click</button><br>
 <p id="one" [hidden]="!button">{{date}}</p>
</form>
```

3. Edit the **app.component.css** as follows:

```
p{color:blue;}
button{color: rebeccapurple;}
#one{color:red;}
```

4. Edit the **app.component.ts** as follows:

```
export class AppComponent {
 title = 'comp';
 date: Date=new Date();button:boolean=false;
 today(){
 this.date=new Date();this.button=!this.button;
}
}
```

Output:

Lab Program 6:

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

• Create different Angular Expressions in Class Component

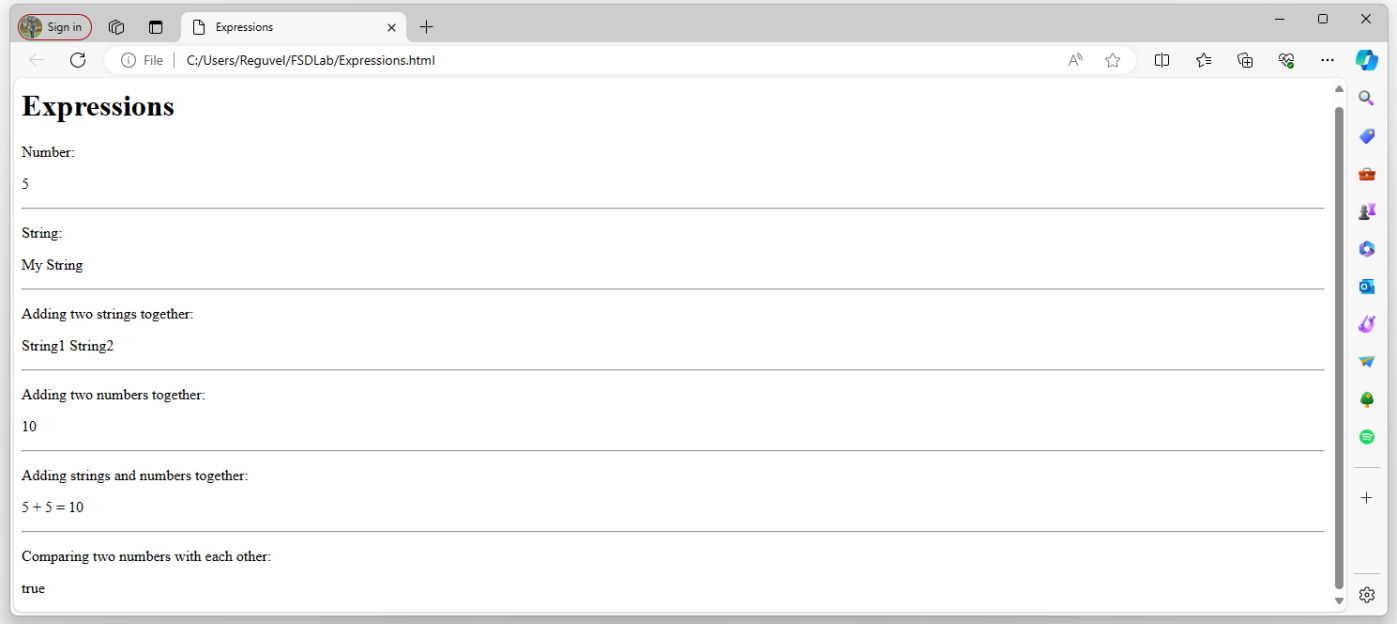
• Also Specify with Different Angular pipes or filters to demonstrate each filter with Angular expression

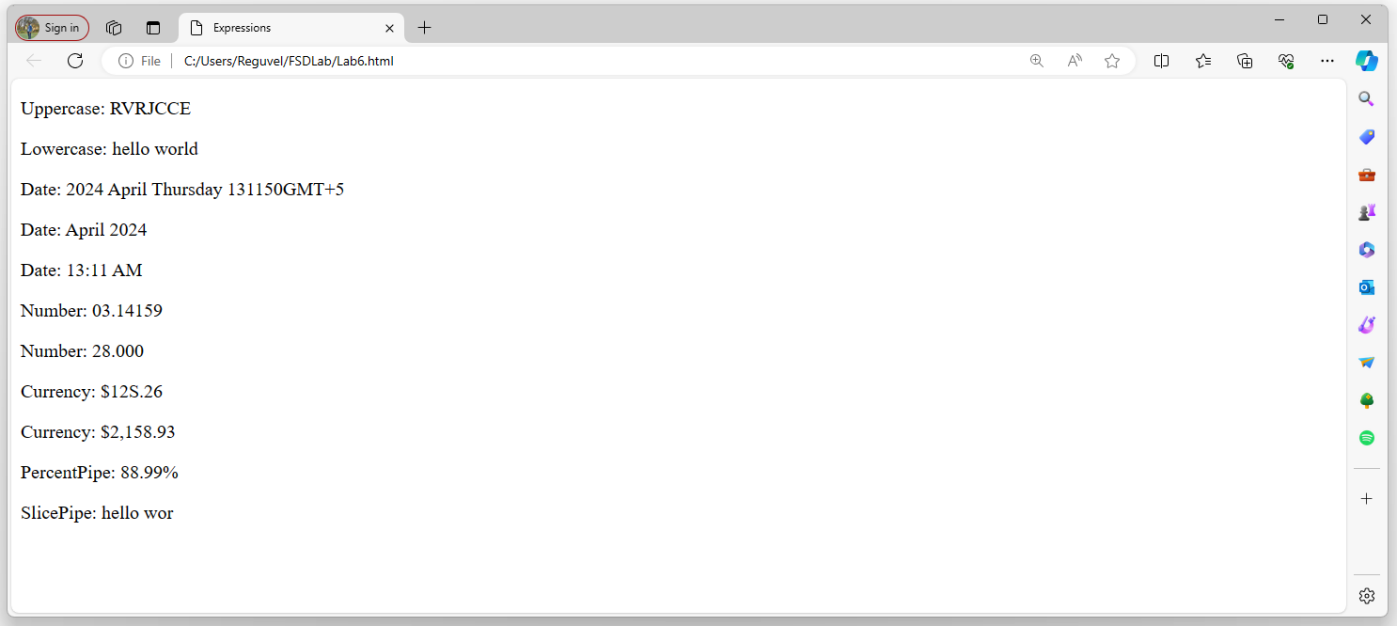
Steps:

Edit inline template and inline style as follows:

```
import { Component } from '@angular/core';
@Component({
selector: 'app-root',
template: `
<h1>Expressions</h1>
Number:<br>
{{5}}<hr>
String:<br>
{{'My String'}}<hr>
Adding two strings together:<br>
{{'String1' + ' ' + 'String2'}}<hr>
Adding two numbers together:<br>
{{5+5}}<hr>
Adding strings and numbers together:<br>
{{5 + '+' + 5 + '='}}{{5+5}}<hr>
Comparing two numbers with each other:<br>
{{5===5}}<hr>
 Uppercase: {{"rvrjcce" | uppercase }}<br>
 Lowercase: {{"HELLO WORLD" | lowercase}}<br>
 Date: {{ today | date:'yMMMMEEEEhmsz'}}<br>
 Date: {{today | date:'mediumDate'}}<br>
 Date: {{today | date: 'shortTime'}}<br>
 Number: {{3.1415927 | number:'2.1-5'}}<br>
 Number: {{28 | number:'2.3'}}<br>
 Currency: {{125.257 | currency:'USD':true: '1.2-2'}}<br>
 Currency: {{2158.925 | currency}}<br>
 PercentPipe: {{.8888 | percent: '2.1'}}<br>
 SlicePipe: {{"hello world" | slice:0:9}}<br> `,
 Styles:[`span{font-weight:bold;border1px ridge-blue;padding:5px}`]
 Export class Pipes{
 Name:String=”RVR”;
 Today:Date;
 Constructor(){
 This.today=new Date()}}
 export class AppComponent {}
```

Output:


Sign in
Expressions
File | C:/Users/Reguvel/FSDLab/Expressions.html
Expressions
Number:
5
String:
My String
Adding two strings together:
String1 String2
Adding two numbers together:
10
Adding strings and numbers together:
5 + 5 = 10
Comparing two numbers with each other:
true


Sign in
Expressions
File | C:/Users/Reguvel/FSDLab/Lab6.html
Uppercase: RVRJCCE
Lowercase: hello world
Date: 2024 April Thursday 131150GMT+5
Date: April 2024
Date: 13:11 AM
Number: 03.14159
Number: 28.000
Currency: $12S.26
Currency: $2,158.93
PercentPipe: 88.99%
SlicePipe: hello wor

Lab Program 7:

Aim:

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

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

Steps:

1. **Write the app.component.html as follows:**

```
<h2>Interpolation Binding</h2>
<p>Welcome {{ name }}</p>
<h2>Style Binding</h2>
<button [style.background-color]="isDisabled ? 'gray' : 'blue'" (click)="change()">Click</button>
<h2>Class Binding</h2>
<div [class.error]="hasError">This text will have error class if hasError is true</div>
<h2>Two-way Binding</h2>
<input type="text" ng-Model="username">
<p>Your username is: {{ username }}</p>
```

2. **In the app.component.ts as rewrite the Appcomponent as follows:**

```
export class AppComponent {
  title = 'bindings';
  name: string = 'mohith';
  isDisabled: boolean = false;
  hasError: boolean = true;
  username: string = '';
  change() {
    this.isDisabled = !this.isDisabled;
  }}
```

3. **Run the using command:**

```
Npm run ng serve
```

Output:

Lab Program 8:

Aim:

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();

Program:

```
var url = require('url');
var urlStr = 'http://user:pass@host.com:80/resource/path?query=string#hash';
var urlObj = url.parse(urlStr,true,false);
urlString = url.format(urlObj);
console.log('Url address: ',urlStr,'\n');
console.log('URL Components:');
console.log('URL Protocol: ',urlObj.protocol);
console.log('URL Host: ',urlObj.host);
console.log('URL port: ',urlObj.port);
console.log('URL Hostname: ',urlObj.hostname);
console.log('URL Path: ',urlObj.path);
console.log('URL Hash: ',urlObj.hash);
var orgUrl = 'http://user.pass@host.com:80/resource/path?query=string#hash';
var newResource = '/another/path?querynew';
console.log('\n',url.resolve(orgUrl,newResource));
```

Output:

```
URL address: http://user:pass@host.com:80/resource/path?query=string#hash

URL Convonents :
URL Protocol: http :
URL Host: host.com:80
URL port: 80
URL Hostname: host.com
URL path: /resource/path?query=string
URL Hash: #hash

http://user.pass@host.com: 80/another/path?querynew
```

Lab Program 9:

Aim:

Implementing Http Server & Client using http node.js module & demonstrate the Http Cli/Ser Application.

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

Steps:

1. Create a folder HTTP and Create a HTML folder to store the HTML file

2. Create a hello.html file in HTML folder:

```
<html><head>
<title>Site</title></head>
<body><h1>Hello World</h1>
<marquee><h1>Reguvel</h1></marquee>
</body></html>
```

3. Create a Client.js and Server.js in HTTP folder

4. Code for Client.js:

```
var http = require('http');
var options = { hostname: '192.168.1.5',
port: '8080', path: '/hello.html' };
function handleResponse(response) {
var serverData = '';
response.on('data', function (chunk) {
serverData += chunk; });
response.on('end', function () {console.log(serverData); }); }
http.request(options, function(response){
handleResponse(response);
}).end();
```

5. Code for Server.js:

```
var fs = require('fs');
var http = require('http');
var url = require('url');
var ROOT_DIR = "html/";
http.createServer(function (req, res) {
var urlObj = url.parse(req.url, true, false);
fs.readFile(ROOT_DIR + urlObj.pathname, function (err,data) {
if (err) {
res.writeHead(404);
res.end(JSON.stringify(err));
return; }
res.writeHead(200);
res.end(data); }); }).listen(8080);
```

6. Run the server in one cmd and run the client in another cmd:

Output:

Lab Program 10:

Aim:

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

Steps:

1. Create a Hello.js file with following code:

```
var express = require('express');
var app = express();
app.get('/', function (req, res) {
 res.send('<h1 style=color:blue;>Hello World</h1>');
})
var server = app.listen(3000, function () {
 var host = server.address().address
 var port = server.address().port
 console.log("Server listening at http://%s:%s", "127.0.0.1", port)
})
```

2. Create a Add.js file with following code:

```
const express = require('express');
const app = express();
app.use(express.urlencoded({ extended: false }));
app.get('/add', (req, res) => {
 res.send(`
<center><h1 style="color:blue">ADDITION</h1>
 <form method="POST" action="/add">
<br><label>Number N1:</label><input type="text" name="t1" placeholder="N1"
required /><br>
<br><label>Number N2:</label><input type="text" name="t2" placeholder="N2"
required /><br>
<br><br>
 <button type="submit">ADD</button>     <button
type="reset">Clear</button>
 </form></center>
 `);
});
app.post('/add', (req, res) => {
 const { t1, t2 } = req.body;
 var n1=Number(t1);
 var n2=Number(t2);
 var result=n1+n2;
 res.send('Addition of Two numbers:'+result+"<br><a style='color:red'
href=./add>GoTo Home</a>");
});
app.listen(3000, () => {
 console.log('Server is running on port 3000');
});
```

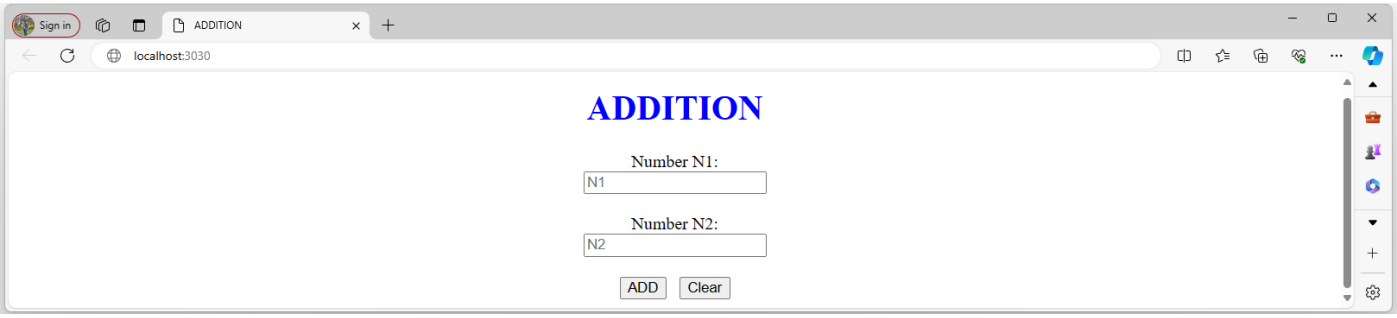
3. Create a Route.js file with following code:

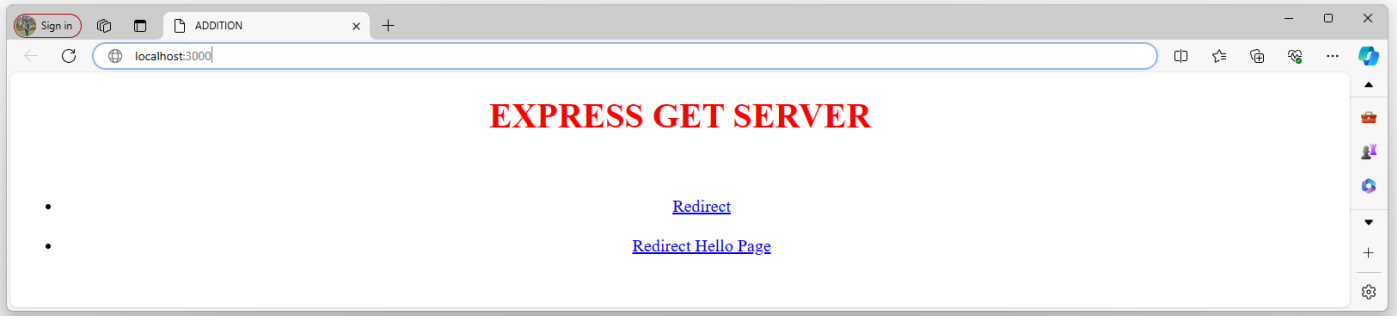
```
var express = require('express');
```

```
var app = express();
app.use(express.urlencoded({ extended: false }));
// This responds with "Hello World" on the homepage
app.get('/', function (req, res) {
res.send})
// This responds a POST request for the homepage
app.get('/express1', function (req, res) {
 res.send('<h1 style=color:red>RED Hello WORLD</h1> <br><a
href=./>GoHome</a>');
})
app.delete('/del_user', function (req, res) {
 res.send('<h1 style=color:red>Hello DELETE</h1>');
})
app.get('/list_user', function (req, res) {
 res.send(`<h1 style=color:blue>Page Listing</h1>
 <a href=./>GoHome</a>`);
})
var server = app.listen(3000, function () {
 var host = server.address().address
 var port = server.address().port
 console.log("Example app listening at http://%s:%s", host, port)
})
```

4. Run each files individually

Output:





Lab Program 11:

Aim:

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.

Steps:

1. Create a login.js file as follows:

```
const express = require('express');
const app = express();
app.use(express.urlencoded({ extended: false }));
app.get('/login', (req, res) => {
console.log("URL:\t " + req.originalUrl);
console.log("Protocol: " + req.protocol);
console.log("IP:\t " + req.ip);
console.log("Path:\t " + req.path);
console.log("Host:\t " + req.host);
console.log("Method:\t " + req.method);
console.log("Query:\t " + JSON.stringify(req.query));
console.log("Fresh:\t " + req.fresh);
console.log("Stale:\t " + req.stale);
console.log("Secure:\t " + req.secure);
console.log("UTF8:\t " + req.acceptsCharset('utf8'));
console.log("Connection: " + req.get('connection'));
console.log("Headers: " + JSON.stringify(req.headers,null,2));

res.send(`
<center><h1 style="color:green">RVRJCCE LOGIN PAGE</h1>
 <form method="POST" action="/login" autocomplete="off">
<br><label>Name:</label><input type="text" name="username"
placeholder="Username" required autooff/><br>
<br><label>Password:</label><input type="password" name="password"
placeholder="Password" required /><br>
<br><br>
 <button type="submit">Login</button>
 </form></center>
 `);
});

app.post('/login', (req, res)=>{
 const { username, password,regd } = req.body;
 if (username === 'reguvel' && password === 'G') {
 res.send('Login successful'+username);
 } else {
 res.send('Invalid username or password:'+username);
 }
});
```

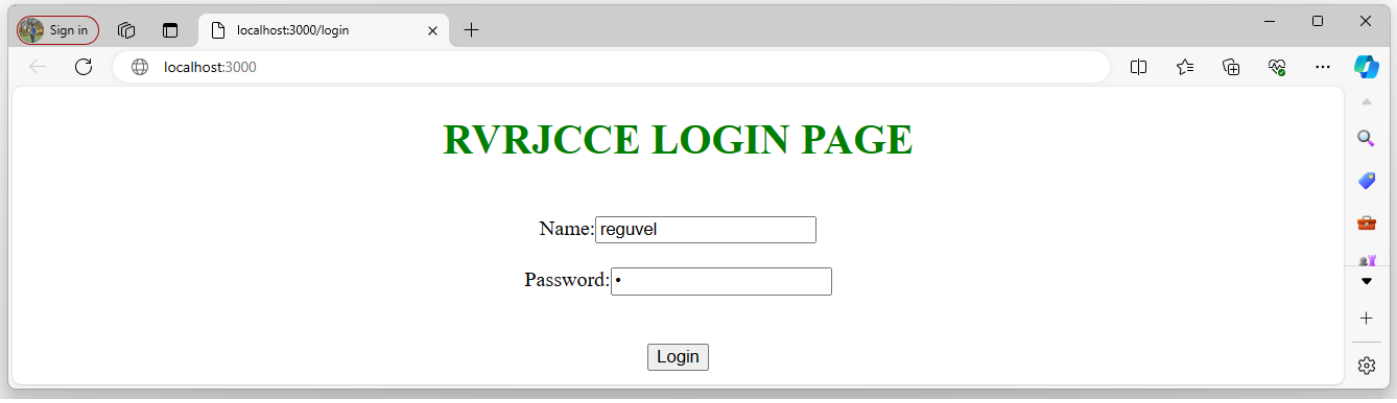
```
app.listen(3000, () => {
 console.log('Server is running on port 3000');
});
```

2. Run the login.js

3. Check the browser in the port 3000

Output:

```
C:\Users\Reguvel\FSDLab\Lab11\node login.js
Server is running on port 3000
```



Terminal Output:

```
D:\STUDY\BTECH\FSD\NodeJS>node login.js
Server is running on port 8080
URL:     /login
Protocol: http
IP:      ::1
Path:    /login
express deprecated req.host: Use req.hostname instead login.js:9:32
Host:    localhost
Method:  GET
Query:   {}
Fresh:   false
Stale:   true
Secure:  false
express deprecated req.acceptsCharset: Use acceptsCharsets instead login.js:15:32
UTF8:    utf8
Connection: keep-alive
Headers: {
  "host": "localhost:8080",
  "connection": "keep-alive",
  "sec-ch-ua": "\"Brave\";v=\"123\", \"Not:A-Brand\";v=\"8\", \"Chromium\";v=\"123\"",
  "sec-ch-ua-mobile": "?0",
  "sec-ch-ua-platform": "\"Windows\"",
  "upgrade-insecure-requests": "1",
  "user-agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.0.0 Safari/537.36",
  "accept":
"text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,/;q=0.8",
  "sec-gpc": "1",
  "accept-language": "en-US,en",
  "sec-fetch-site": "none",
  "sec-fetch-mode": "navigate",
  "sec-fetch-user": "?1",
  "sec-fetch-dest": "document",
  "accept-encoding": "gzip, deflate, br, zstd"
}
```

Lab Program 12:

Aim:

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

Steps:

1. **Install MongoDb Community edition**
2. **Open MongoDb Compass,start the server**
3. **Install mongodb in node.js by the command:**

```
npm install mongodb –save
```

4. **Create a file named mongo.js as follows:**

```
const { MongoClient } = require("mongodb");
const uri = "mongodb://localhost:27017";
const dbName = "mydatabase";
async function main() {
const client = new MongoClient(uri);
try {
await client.connect();
console.log("Connected to MongoDB server");
const db = client.db(dbName);
await db.createCollection("students");
await db.collection("students").insertOne({
name: "Gnanam",
age: 50,
Rgno: "1978", });
await db.collection("students").insertOne({
name: "Reguvel",
age: 20,
Rgno: "Y21CS043", });
await db
.collection("students")
.updateOne({ name: "Reguvel" }, { $set: { Rgno: "Y21CS43" } });
await db.collection("students").deleteOne({ name: "Satuluri" });
} finally {
await client.close(); } }
main().catch(console.error);
```

Output: