FULL STACK DEVELOPMENT- CS325

(LAB RECORD)

RVR & JC COLLEGE OF ENGINEERING

CHOWDAVARAM, GUNTUR-522 019 (AUTONOMOUS)

Affiliated to ACHARYA NAGARJUNA UNIVERSITY :: GUNTUR -10

p.Siva Prasad
Lecturer Incharge

Dr.M.Sreelatha
Prof.&HOD,Dept.of CSE

By :
U.Mahesh Reddy
(Y21CS181)

CS325-1(JOEL02)::LBD Course R-20 ::

FULL STACK DEVELOPMENT

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();	13
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	14
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	16
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.	19
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	21

Lab 01

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

Steps:

1.Installation of Node.js Environment:

1. Go to the official Node.js website: https://nodejs.org/
2. Download the appropriate installer for your operating system.
3. Follow the installation instructions provided on the website.
4. Once installed, you can verify the installation by opening your terminal or command prompt and typing:node -v(This command will display the installed Node.js version.)

2.Testing through the Node REPL Shell:

1. Open your terminal or command prompt.
2. Type node and press Enter to start the Node.js REPL shell.
3. You can now execute JavaScript commands directly in the shell. For example:> console.log("Hello, Node.js!")
4. To exit the REPL shell, press Ctrl + C twice or type .exit and press Enter.

3.Installing prompt-sync Module:

1. Open your terminal or command prompt.
2. Navigate to your project directory or any directory where you want to install the module.
3. Run the following command to install prompt-sync globally: npm install -g prompt-sync
4. Alternatively, if you want to install prompt-sync locally for a specific project, navigate to your project directory and run: npm install prompt-sync

4.Testing prompt-sync with Console Module Application:

1. Create a new JavaScript file (e.g., app.js) in your project directory by entering notepad <filename> in cmd
2. Write some code to test the prompt-sync module. For example:

```
const prompt = require('prompt-sync')();

const name = prompt('Enter your name: ');

console.log('Hello, ' + name + '!');
```

3. Save the file.
4. Open your terminal or command prompt.
5. Navigate to the directory where your app.js file is located.f. Run the application by typing: node app.js

OUTPUT:

```
Your environment has been set up for using Node.js 20.11.1 (x64) and npm.
C:\Users\MaheshReddy>cd fsd
C:\Users\MaheshReddy\FSD>node -v
v20.11.1
C:\Users\MaheshReddy\FSD>node
Welcome to Node.js v20.11.1.
Type ".help" for more information.
> console.log("Hello, Node.js!")
Hello, Node.js!
undefined
>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
>
C:\Users\MaheshReddy\FSD>npm install -g prompt-sync
changed 3 packages in 319ms
C:\Users\MaheshReddy\FSD>npm install prompt-sync
up to date, audited 4 packages in 692ms
found 0 vulnerabilities
C:\Users\MaheshReddy\FSD>notepad app.js
C:\Users\MaheshReddy\FSD>node app.js
Enter your name: MaheshReddy
Hello, MaheshReddy!
```

Lab 02

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

Steps:

1. Create a folder name date

2. Create a mydate.js file which exports the current date and time:

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

3. Create a date.js file to import mydate.js file and display the date and time:

```
const myDate = require('./mydate.js');
const currentDate = myDate.getCurrentDate();
console.log('Current Node.js Webserver time and date:', currentDate);
```

4.Create package.json file:

```
{
  "name": "my-date-module",
  "version": "1.0.0",
  "description": "A custom Date module for Node.js",
  "main": "date.js",
  "author": "MaheshReddy",
  "scripts": {"test": "echo \"Error: no test specified\" && exit 1"},
  "license": "ISC"
}
```

5. Enter npm publich command to publish

OUTPUT:

```
C:\Users\MaheshReddy\FSD\lab\date>node date.js
Current Node.js Webserver time and date: 2024-04-16T13:59:04.041Z
```

Lab 03

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.

Steps:

1.Create a new directory for your project:

mkdir welcome-node-app

2. Navigate into the project directory:

cd welcome-node-app

3. Initialize npm in the project directory:

npm init -y

4. Create a folder structure for your application:

mkdir src

5.Inside the src directory, create a JavaScript file for your application:

cd src

notepad app.js

6.Open the app.js file in your preferred code editor (e.g., Visual Studio Code):

```
const myDate = require('./myDate');


function greet() {
   const currentDate = myDate.getCurrentDate();
   console.log(`Welcome to the Node.js App! Today's date is ${currentDate}`);
}
greet();
```

7. Create a custom module myDate.js inside the src directory:

notepad myDate.js

```
function getCurrentDate() {
   return new Date().toDateString();}
module.exports = {
   getCurrentDate};
```

8. Go back to the root directory of your project:

cd ..

9. Run your Node.js application:

node src/app.js

OUTPUT:

```
C:\Users\MaheshReddy\FSD\lab>mkdir welcome


C:\Users\MaheshReddy\FSD\lab>cd welcome


C:\Users\MaheshReddy\FSD\lab\welcome>npm init -y
Wrote to C:\Users\MaheshReddy\FSD\lab\welcome\package.json:
{
  "name": "welcome",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
C:\Users\MaheshReddy\FSD\lab\welcome>mkdir src
C:\Users\MaheshReddy\FSD\lab\welcome>cd src
C:\Users\MaheshReddy\FSD\lab\welcome\src>notepad app.js
C:\Users\MaheshReddy\FSD\lab\welcome\src>notepad myDate.js
C:\Users\MaheshReddy\FSD\lab\welcome\src>cd ..
C:\Users\MaheshReddy\FSD\lab\welcome>node src/app.js
Welcome to the Node.js App! Today's date is Tue Apr 16 2024
```

Lab 04

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 messag

Steps:

1. Create a new Angular project with Angular CLI:

ng new comp

select CSS and <enter>, enter “y” for yes

2. Navigate into the project directory:

cd comp

3. Go to comp/src/app

4. Edit app.component.ts as follows:

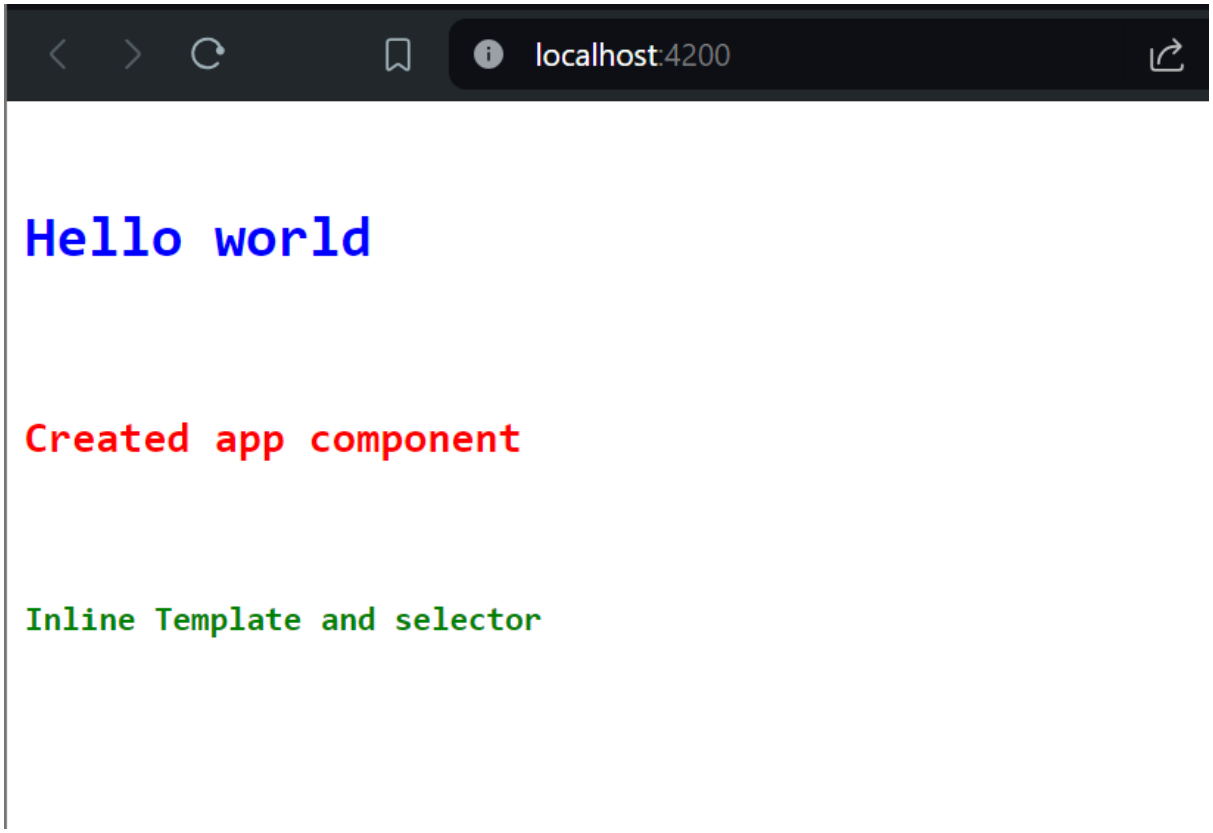
```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
@Component({
 selector: 'app-root',
 standalone: true,
 imports: [RouterOutlet],
 template: `
<pre>
<h1 >Hello world</h1>
<h2 >Created app component</h2>
<h3 >Inline Template and selector</h3></pre>`,
 styles:[`h1{color:blue;}h2{color:red;} 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 05

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>
```

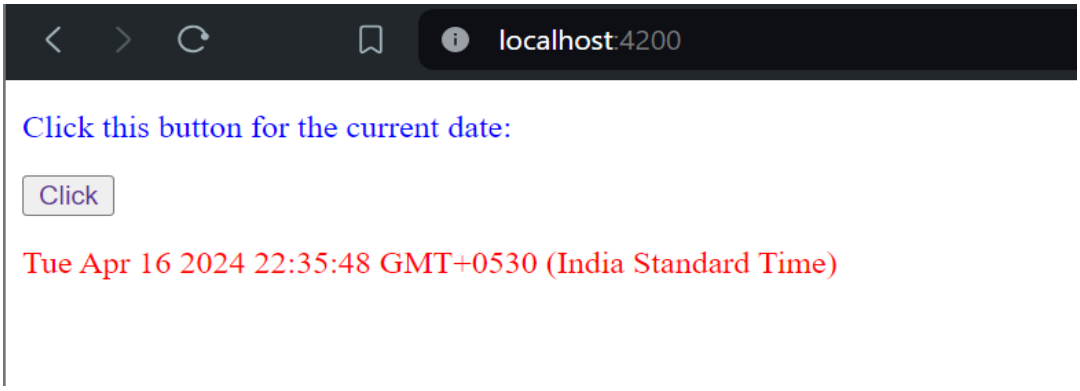
3.Edit the app.component.css as follows:

```
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 06

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:

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

Uppercase: RVRJCCE
Lowercase: hello world
Date: 2023MayMonday101840GMT+5
Date: May 8, 2023
Date: 10:18 AM
Number: 03.14159
Number: 28.000
Currency: $125.26
Currency: $2,158.93
PercentPipe: 88.9%
SlicePipe: hello wor

Lab 07

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 = 'MaheshReddy';

 isDisabled: boolean = false;

 hasError: boolean = true;

 username: string = '';

 change() {

  this.isDisabled = !this.isDisabled;

 }}
```

3. Run the using command:

```
Npm run ng serve
```

OUTPUT:

Interpolation Binding

Welcome syam

Style Binding

Click

Class Binding

This text will have error class if hasError is true

Two-way Binding

Mahesh Reddy

Your username is: Mahesh Reddy

Lab 08

AimCreate 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:'+urlstring);
console.log('url components');
console.log('protocol:'+urlobj.protocol);
console.log('host:'+urlobj.host);
console.log('auth:'+urlobj.auth);
console.log('port:'+urlobj.port);
console.log('hostname:'+urlobj.hostname);
console.log('path:'+urlobj.path);
console.log('hash:'+urlobj.hash);
var orginalurl='http://user:pass@host.com:80/resource/path?query=string#hash';
var newresource='/another/path?querynew';
console.log(url.resolve(orginalurl,newresource));
```

OUTPUT:

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

url components

protocol:http:

host:host.com:80

auth:user:pass

port:80

hostname:host.com

path:/resource/path?query=string

hash:#hash

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

Lab 09

AimImplementing 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

Steps:

1. Create a folder HTTP

2. Create a HTML folder to store the HTML file

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

```
<html><head>
<title>Paaaaa</title>
</head>
<body>
<h1>Hello World</h1>
<marquee><h1>MaheshReddy</h1></marquee>
</body></html>
```

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

5. Code for Client.js:

```
var http = require('http');
 var options = {
 hostname: '192.168.1.5',
 port: '8095',
 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();
```

6. 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(8095);
```

7. Run the server in one cmd and run the client in another cmd

OUTPUT:

Hello World

Mahesh Reddy

Lab 10

AimCreate 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');

});
```

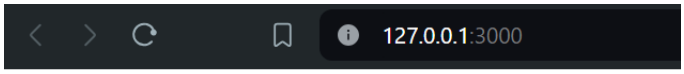
3. Create a Route.js file with following code:

```
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(`<center><h1 style=color:red>EXPRESS GET SERVER</h1><br><ul><li><a
href=./list_user>Redirect</a>

</li><br><li><a href=./express1>Redirect Hello Page</a></li></ul></center>`);

})

// 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>');

})
```

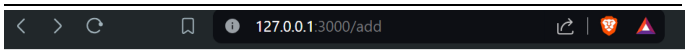
```
app.delete('/del_user', function (req, res) {
  res.send('<h1 style=color:red>Hello DELETE</h1>');
})
app.get('/list_user', function (req, res) {
  res.send(`<h1 style=color:blue>Page Listing</h1>
<a href=./>GoHome</a>`);
})
var server = app.listen(3000, function () {
  var host = server.address().address
  var port = server.address().port
  console.log("Example app listening at http://%s:%s", host, port)
})
```

4. Run each files individually

OUTPUT:



Hello World



ADDITION

Number N1: 10

Number N2: 20

ADD Clear

EXPRESS GET SERVER

- Redirect
- Redirect Hello Page

Lab 11

AimCreate 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:

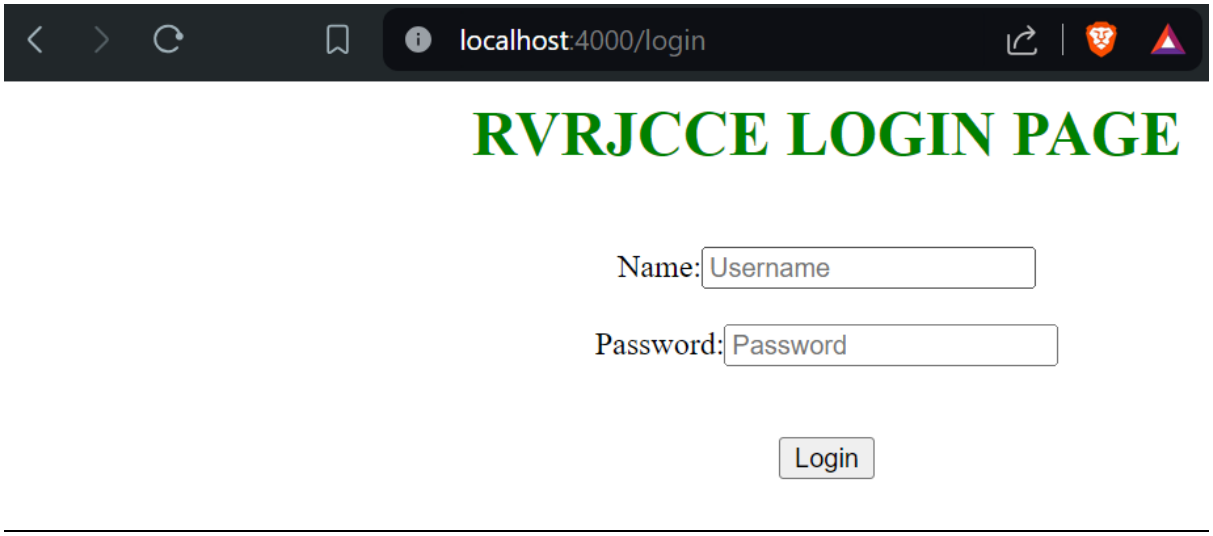
```
const express = require('express');

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 === 'MaheshReddy' && password === 'Gollamandala') {

  res.send('Login successful'+username);

 } else {

  res.send('Invalid username or password:'+username);

 }

});

app.listen(4000, () => {

 console.log('Server is running on port 4000');

});
```

2. Run the login.js

3. Check the browser in the port 4000

OUTPUT:



Lab 12

AimCreate 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 from the link : https://www.mongodb.com/try/download/community

2. Install MongoShell from the link : https://www.mongodb.com/try/download/shell

3. Open MongoDb Compass,start the server

4. Mongosh commands:

```
use student // to use the student database

db.student.InsertOne(({name: "MaheshReddy",age: 20,Rgno: "46"});// to insert one

db.student.InsertMany[({name: "Teja",age: 20,Rgno: "21"},({name: "Pavan",age: 20,Rgno: "63"},({name: "MaheshReddy",age: 20,Rgno: "46"}];// to insert many

db.student.updateOne({ name: "MaheshReddy" },{ $set: { Rgno: "Y21CS046" } });// to update one

db.student.updateMany();// to update many

db.student.deleteMany();// to delete many

db.student.deleteOn();// to delete one
```

5. Install mongodb in node.js by the command:

```
npm install mongodb –save
```

6. 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: "MaheshReddy",
      age: 20,
      Rgno: "46"
    });
    await db.collection("students").insertOne({
      name: "Gollamandala",
      age: 20,
      Rgno: "Y21CS046"
    });
    await db.collection("students").updateOne(
      { name: "MaheshReddy" },
      { $set: { Rgno: "Y21CS046" } }
    );
    await db.collection("students").deleteOne({ name: "Gollamandala" });
  } finally {
    await client.close();
  }
}
main().catch(console.error);
```

OUTPUT: