|  |  |
| --- | --- |
| **EX NO: 8-(d)** | **CRUD OPERATIONS USING MONGODB, NODEJS AND HTML** |
| **DATE:15/04/2024** |

**AIM:**

To perform crud operations using mongodb,nodejs and html.

**PROCEDURE:**

* Initialize Express and MongoDB: Import the required modules and initialize an Express application.
* Middleware: Use cors and body-parser middleware for handling CORS and parsing request bodies.
* Database Check: In the root route handler, connect to MongoDB and check if the database exists.
* Collection Check: If the database exists, check if the collection “Employee” exists.
* Collection Creation: If the collection doesn’t exist, create it.
* Employee Search: In the “/employeesearch” route handler, connect to MongoDB, query the “Employee” collection for the provided name, and return the result.
* Employee Insertion: In the “/employeeinsert” route handler, connect to MongoDB, insert a new document into the “Employee” collection with the provided name, department, and date of joining, and return a success message.
* Employee Updation: In the “/employeeupdate” route handler, connect to MongoDB, update a new document into the “Employee” collection with the provided name, department, and date of joining, and return a success message.
* Employee Deletion: In the “/employeedel” route handler, connect to MongoDB, delete the document from the “Employee” collection with the provided name, and return a success message.

**PROGRAM:**

**Employee.js:**

const express=require('express')

const {MongoClient} = require('mongodb');

const bodyParser=require('body-parser');

const app=express();

const cors=require('cors')

app.use(cors());

app.use(express.json())

app.use(bodyParser.urlencoded({

extended:true

}))

app.post("/", async function(req, res){

MongoClient.connect(url, function(err, client) {

if (err) throw err;

var db = client.db("FSD");

// Check if the database exists

db.admin().listDatabases((err, dbs) => {

if (err) throw err;

const databaseExists = dbs.databases.some(db => db.name === "FSD");

if (databaseExists) {

console.log("Database already exists!");

// Check if the collection exists

db.listCollections({ name: "Employee" }).toArray((err, collections) => {

if (err) throw err;

if (collections.length > 0) {

console.log("Collection already exists!");

} else {

// Create the collection

db.createCollection("Employee", function(err, res) {

if (err) throw err;

console.log("Collection created!");

});

}

});

} else {

console.log("Database does not exist.");

}

});

});

});

app.post("/employeesearch", async function(req, res) {

try {

let word = req.body.name;

console.log("Input: " + word);

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'FSD';

await client.connect();

const db = client.db(database);

const query = { name: word };

const collection = db.collection('Employee');

const response = await collection.find(query).toArray();

if (response.length === 0) {

// If no data is found, send an appropriate message

throw new Error("Data not found.");

}

console.log(response);

res.send("<h1>Name of the Employee: " + word + "<br>Department: " + response[0].Dept + "</h1>");

} catch (error) {

console.error("Error:", error);

res.status(404).send("Data not found.");

}

});

app.post("/employeeinsert", async function(req, res1) {

try {

const word = req.body.name;

const dept = req.body.Dept;

const doj = req.body.DOJ;

console.log("Input: " + word + ", " + dept + ", " + doj);

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'FSD';

await client.connect();

const db = client.db(database);

const document = { name: word, Dept: dept, DOJ: doj };

const collection = db.collection('Employee');

const result = await collection.insertOne(document);

if (result.insertedCount === 0) {

throw new Error("Failed to insert data.");

}

console.log("Data inserted successfully");

res1.send("<h1>Data Inserted successfully</h1>");

} catch (error) {

console.error("Error:", error);

res1.status(500).send("Failed to insert data.");

}

});

app.post("/employeedel", async function(req, res1) {

try {

const word = req.body.name;

console.log("Input: " + word);

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'FSD';

await client.connect();

const db = client.db(database);

const query = { name: word };

const collection = db.collection('Employee');

const result = await collection.deleteOne(query);

if (result.deletedCount === 0) {

// If no document was deleted, it means the data wasn't found

throw new Error("Data not found for deletion.");

}

res1.send("<h1>Data deleted successfully</h1>");

} catch (error) {

console.error("Error:", error);

res1.status(404).send("Data not found for deletion.");

}

});

app.post("/employeeupdate", async function(req, res1) {

try {

const word = req.body.name;

const dept = req.body.Dept;

const doj = req.body.DOJ;

console.log("Input: " + word);

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'FSD';

await client.connect();

const db = client.db(database);

const query = { name: word };

const newValues = { $set: { name: word, Dept: dept, DOJ: doj} };

const collection = db.collection('Employee');

const result = await collection.updateOne(query, newValues);

if (result.matchedCount === 0) {

throw new Error("Data not found for update");

}

console.log(result);

res1.send("<h1>Data updated successfully</h1>");

} catch (error) {

console.error("Error:", error);

res1.status(500).send("An error occurred while updating data: " + error.message);

}

});

app.listen(8020,function(){

console.log("Server is running on port number 8020")

});

**Index.html:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Navigation Bar</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}

nav {

background-color: #333;

color: #fff;

display: flex;

justify-content: center;

padding: 10px 0;

}

nav a {

color: #fff;

text-decoration: none;

padding: 0 15px;

}

nav a:hover {

text-decoration: underline;

}

.form-container {

display: none;

justify-content: center;

margin-top: 20px;

}

.form-container.active {

display: flex;

}

.form-container input[type="text"] {

padding: 8px;

margin-right: 5px;

}

.form-container button {

padding: 8px 15px;

background-color: #4CAF50;

color: white;

border: none;

cursor: pointer;

}

.form-container button:hover {

background-color: #45a049;

}

</style>

</head>

<script>

</script>

<body>

<nav>

<a href="#" onclick="toggleForm('create')">Create</a>

<a href="#" onclick="toggleForm('insert')">Insert</a>

<a href="#" onclick="toggleForm('update')">Update</a>

<a href="#" onclick="toggleForm('search')">Search</a>

<a href="#" onclick="toggleForm('delete')">Delete</a>

</nav>

<!-- Create Form -->

<div id="create-form" class="form-container">

<form method="post">

<input type="text" name="name" id="name" placeholder="DB Name">

<button>Create</button>

</form>

</div>

<!-- Insert Form (similar structure for other forms) -->

<div id="insert-form" class="form-container">

<form action="http://localhost:8020/employeeinsert" method="post">

<input type="text" name="name" id="name" placeholder="Name">

<input type="text" name="Dept" id="Dept" placeholder="Department">

<input type="text" name="DOJ" id="DOJ" placeholder="Date of Joining">

<button type="submit">Insert</button>

</form>

</div>

<!-- Update Form -->

<div id="update-form" class="form-container">

<form action="http://localhost:8020/employeeupdate" method="post">

<input type="text" name="name" id="name" placeholder="The Name to be updated">

<input type="text" name="Dept" id="Dept" placeholder="Department">

<input type="text" name="DOJ" id="DOJ" placeholder="Date of Joining">

<button type = "submit">Update</button>

</form>

</div>

<!-- Search Form -->

<div id="search-form" class="form-container">

<form action="http://localhost:8020/employeesearch" method="post">

<input type="text" name="name" id="name" placeholder="Name">

<button>Search</button>

</form>

</div>

<!-- Delete Form -->

<div id="delete-form" class="form-container">

<form action="http://localhost:8020/employeedel" method="post">

<input type="text" name="name" id="name" placeholder="Name">

<button>Delete</button>

</form>

</div>

<script>

function toggleForm(formId) {

var forms = document.getElementsByClassName("form-container");

for (var i = 0; i < forms.length; i++) {

forms[i].classList.remove("active");

}

document.getElementById(formId + "-form").classList.add("active");

}

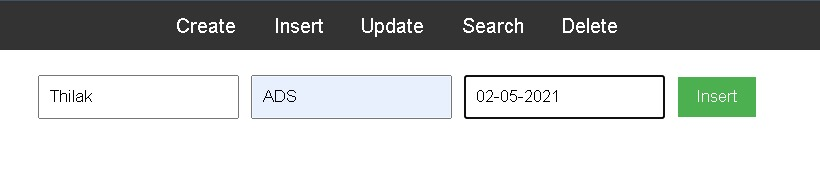
</script>

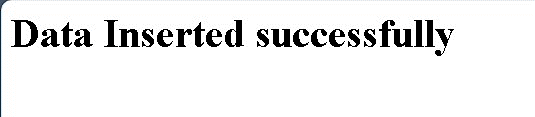
</body>

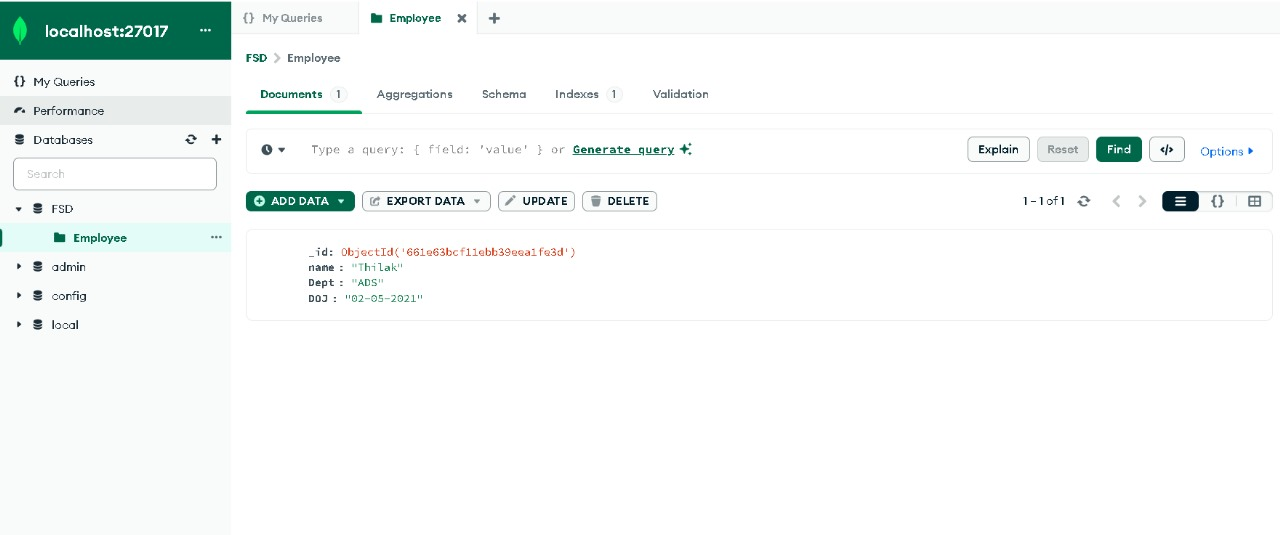
</html>

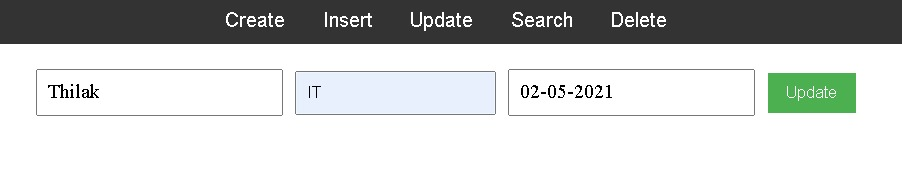
**OUTPUT:**

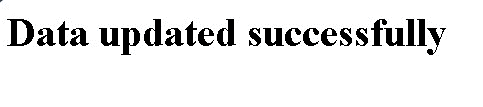
**Insert:**

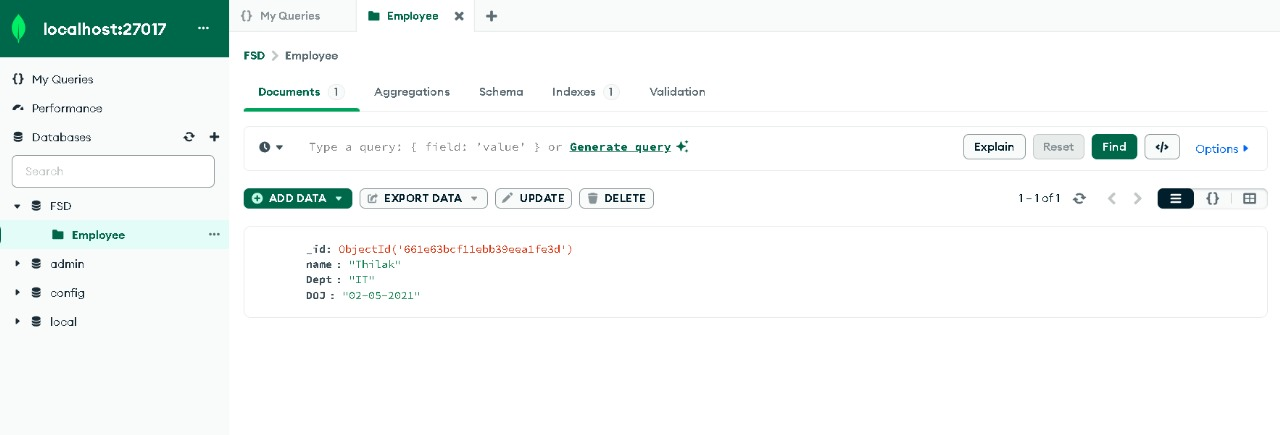




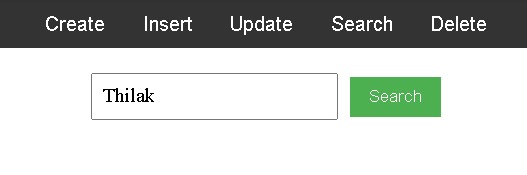
**Update:**







**Search:**

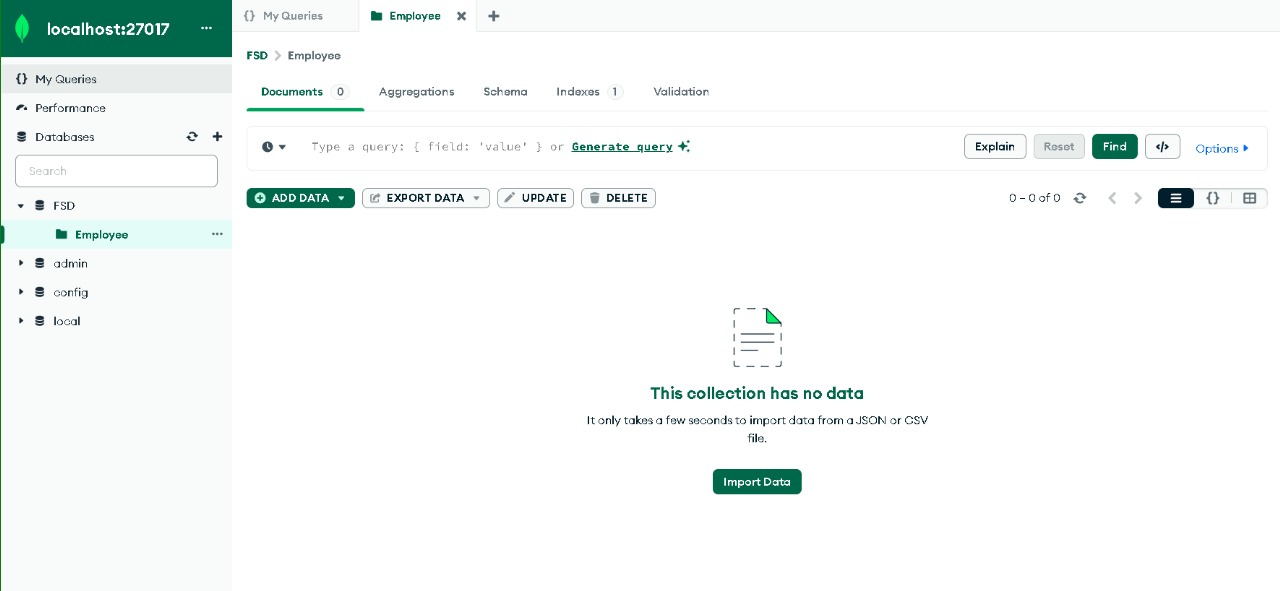




Delete:







**RESULT:**

Hence the program to execute a CRUD operations in mongodb using node js and html is executed successfully.

|  |  |
| --- | --- |
| **EX NO: 9-(a)** | **SENDING MAIL USING NODEJS(NODEMAILER)** |
| **DATE:15/04/2024** |

**AIM:**

To send a mail using nodemailer module in node js.

**PROCEDURE:**

* Initialize Express and Nodemailer: Import the required modules and initialize an Express application. Also, create a transport for Nodemailer with the appropriate service and authentication details.
* Middleware: Use cors and body-parser middleware for handling CORS and parsing request bodies.
* Compute Marks Route: In the “/computeMarks” route handler, retrieve the marks from the request body, calculate the total marks, determine the result, and calculate the average.
* Create Message: Create an HTML message that includes a table with the email, registration number, result, and average.
* Send Email: Create mail options with the sender’s email, recipient’s email, subject, and HTML message. Use the transporter to send the email.
* Send Response: Send the HTML message as the response to the client.
* Start Server: Start the Express server on port 5000.

**PROGRAM:**

const express=require('express')

const bodyParser=require('body-parser');

const nodemailer=require('nodemailer');

const app=express();

const cors=require('cors')

app.use(cors());

app.use(express.json())

app.use(bodyParser.urlencoded({

extended:true

}))

var transporter = nodemailer.createTransport({

service: 'Outlook365',

auth: {

user: xxxx@sonatech.ac.in',

pass: 'xxxx'

},

tls : { rejectUnauthorized: false }

});

app.post("/computeMarks",function(req,res){

let regno=req.body.regno;

let email=req.body.email;

let machinelearning=req.body.machinelearning;

let fullstackdevelopment=req.body.fullstackdevelopment;

let agile=req.body.agile;

let totalqualitymanagement=req.body.totalqualitymanagement;

let mllab=req.body.mllab;

let fsdlab=req.body.fsdlab;

let totalmarks=machinelearning+fullstackdevelopment+agile+totalqualitymanagement+mllab+fsdlab;

let result=(machinelearning>=50 && fullstackdevelopment>=50 && agile>=50 && totalqualitymanagement>=50 && mllab>=50 && fsdlab>=50)?"pass":"fail";

let avg=totalmarks/6;

let message = `

<table style="width:100%; border: 1px solid black;">

<tr>

<th>Email</th>

<th>Reg No.</th>

<th>Result</th>

<th>Average</th>

</tr>

<tr>

<td>${email}</td>

<td>${regno}</td>

<td>${result}</td>

<td>${avg}</td>

</tr>

</table>

`;

var mailOptions = {

from: xxxx@sonatech.ac.in',

to: email,

subject: 'Semester Result',

html: message

}

transporter.sendMail(mailOptions, function(error, info){

if (error) {

console.log(error);

} else {

console.log('Email sent: ' + info.response);

}

});

res.send(message)

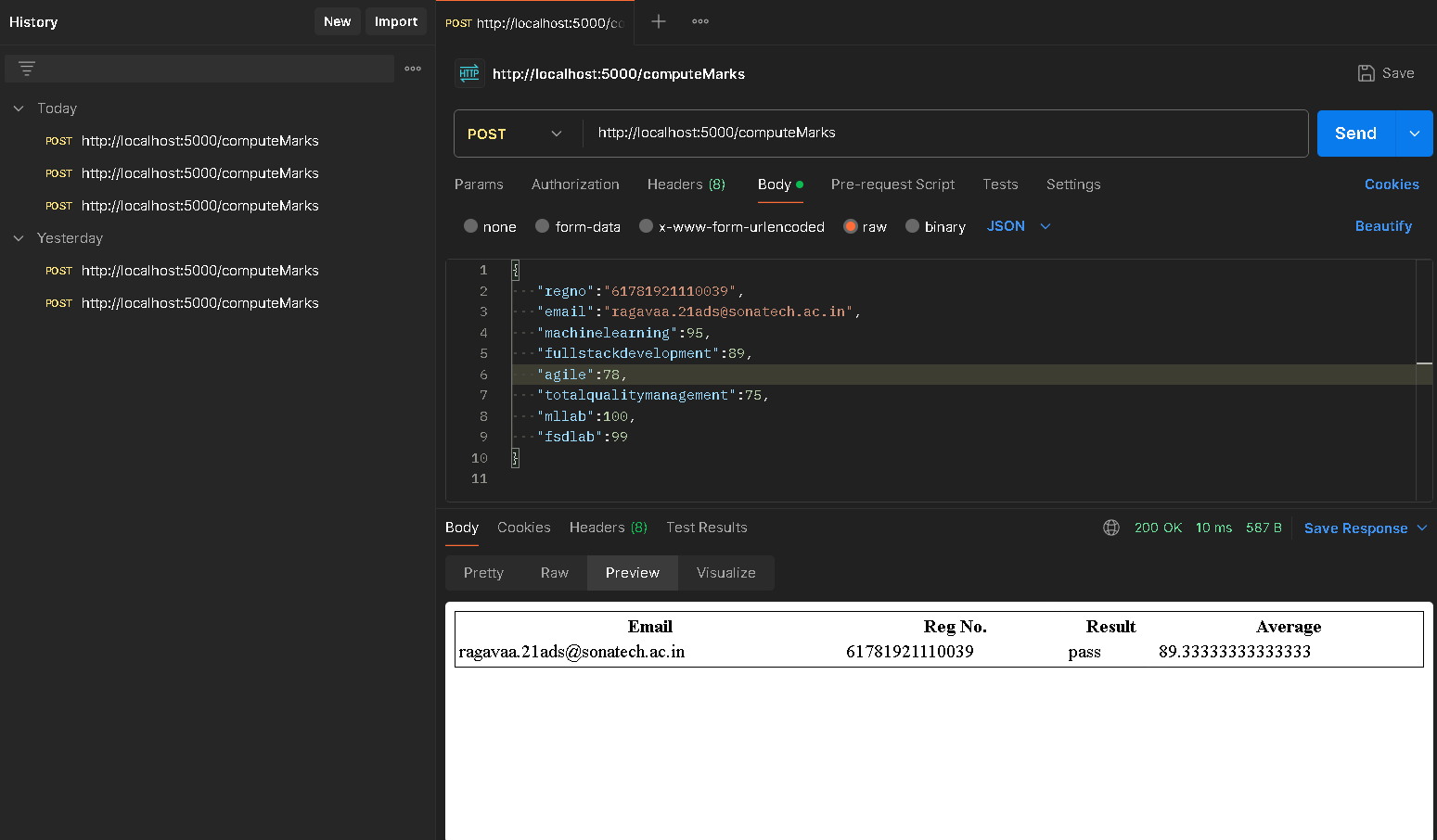
})

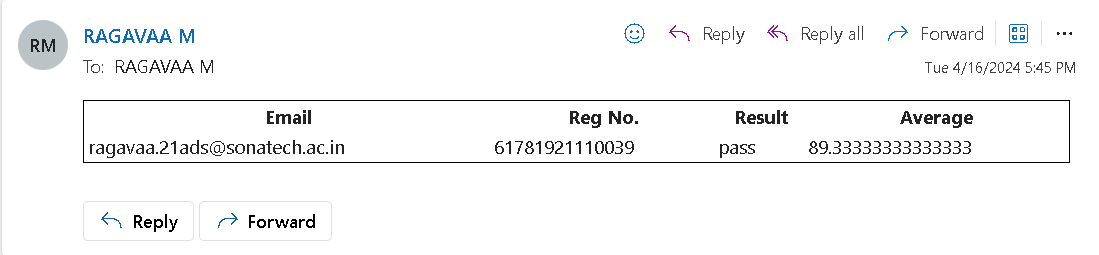
app.listen(5000,function(){

console.log("Server is running on port number 5000")

})

**OUTPUT:**





**RESULT:**

Hence the program to send a mail using nodemailer module in node js is executed successfully.

|  |  |
| --- | --- |
| **EX NO: 9-(b)** | **ASSESMENT USING REACT JS AND NODEMAILER** |
| **DATE:15/04/2024** |

**AIM:**

To perform a assessment using react js , mongodb and mailing the secured marks through nodemailer .

**PROCEDURE:**

1. Goto the terminal in visual code and execute npx create-react-app my-http

2. Install the following dependency files

* npm install axios react-bootstrap react-dom react-router dom.

3. go to the public folder and edit the index.js file and add the following code insde head of the HTML web page.

4. Create folder called "images" inside "public" folder and copy the images that are required to be shown in careousl (See the attachement)

5. Inside the src folder,create a folder called Components and create the following js :

Header.js, Car.js, Signup.js, Question.js and Login.js

6. Update the index.js file inside the src folder.

7. Create folder called backend in visual code terminal in the same project folder.

8. Change to the "backend" folder and execute the following command

* npm init and accept the default settings

9. Install the following command to install all dependency files

* /backend> npm install body-parser cors express mongodb nodemailer

10. Create index.js file.

11. Run the index.js file in backend

* node index.js

12. You can start interacting with the front-end

13. Run the front-end i.e go to the reactjs terminal and type the following command

* npm start

**PROGRAM:  
  
src/index.js:**

import React from 'react'

import ReactDOM from 'react-dom/client'

import { BrowserRouter as Router,Routes, Route, Link } from 'react-router-dom';

import Car from './Components/Car';

import Signup from './Components/Signup';

import Login from './Components/Login';

import Header from './Components/Header';

const root=ReactDOM.createRoot(document.getElementById('root'));

root.render(

<Router>

<div>

<Header/>

<Routes>

<Route exact path="/" element={<Car/>}></Route>

<Route exact path="/Signup" element={<Signup/>}></Route>

<Route exact path="/Login" element={<Login/>}></Route>

</Routes>

</div>

</Router>

)

**src/components/car.js:**import React from 'react';

import 'bootstrap/dist/css/bootstrap.css';

import 'bootstrap/dist/css/bootstrap.min.css';

import Carousel from 'react-bootstrap/Carousel';

export default function Car() {

const images=["/images/image 1.png",

"/images/image 2.jpg",

"/images/image 3.png"]

return (

<div claasName="container-fluid" style={{display: 'block', height:90, width: 900, paddingTop:1,marginLeft:160}}>

<Carousel>

<Carousel.Item interval={1500}>

<img

className="d-block w-100"

src={images[0]}

alt="Image One"

/>

<Carousel.Caption>

<h3>First Slide</h3>

<p>First Image</p>

</Carousel.Caption>

</Carousel.Item>

<Carousel.Item interval={500}>

<img

className="d-block w-100"

src={images[1]}

alt="Image Two"

/>

<Carousel.Caption>

<h3>Second slide</h3>

<p>Second Image</p>

</Carousel.Caption>

</Carousel.Item>

<Carousel.Item interval={500}>

<img

className="d-block w-100"

src={images[2]}

alt="Image Two"

/>

<Carousel.Caption>

<h3>Third slide</h3>

<p>Third Image</p>

</Carousel.Caption>

</Carousel.Item>

</Carousel>

</div>

)

}

**src/Components/header.js:**  
import React from 'react'

export default function Header() {

return (

<div>

<nav className="navbar navbar-expand-xl bg-dark justify-content-center">

<div className="container-fluid">

<ul className="nav navbar-nav">

<li className="active">

<a href="/" style={{textDecoration:'none',color:'white'}}> <span className="display-4">Home </span></a></li>

&nbps;

<li><a href="/Signup" style={{textDecoration:'none',color:'white'}}><span className="display-4">Signup </span></a></li>

&nbps;

<li><a href="/Login" style={{textDecoration:'none',color:'white'}}><span className="display-4">Login </span></a></li>

</ul>

</div>

</nav>

</div>

)

}

**src/Components/Login.js:**  
import React, { Component } from 'react'

import axios from 'axios'

import Questions from './Question'

export default class Login extends Component

{

constructor(props) {

super(props)

this.state = {

username:"",

password:"",

SigninDetails:'',

error:''

}

this.allinone=this.allinone.bind(this)

this.handleLogin=this.handleLogin.bind(this)

}

allinone(event)

{

this.setState({[event.target.name]:event.target.value})

console.log(event.target.value)

}

handleLogin(event)

{

console.log(this.state)

axios.post("http://localhost:5000/signin",this.state)

.then (res=>{

console.log(res.data)

this.setState({SigninDetails:res.data})

})

.catch(error=>{

console.log("Data Loding Error")

this.setState({err:"Data Loding Error"})

})

}

render() {

return (

<div>

{this.state.SigninDetails==="success"

?<div>

{console.log(this.state.SigninDetails)}

<h1 className='text-success'> Hai {this.state.username}, Login Sucessfull,

</h1>

<Questions username={this.state.username}/>

</div>

:null

}

{this.state.SigninDetails==="Failure"

?

<div>

{console.log(this.state.SigninDetails)}

<h1 className='text-danger'> Invalid Username or Password</h1>

<form>

<div className="form-group">

<label>Username (Preferrably Your E-Mail Id):</label>

<input style={{width:'40%'}} type="text" className="form-control" id="username" placeholder="Enter Name" name="username" onChange={this.allinone}/>

</div>

<div className="form-group">

<label>Password:</label>

<input style={{width:'40%'}} name="password" type="password" className="form-control" id="password" placeholder="Enter password" name="password" onChange={this.allinone} />

</div>

<button type="button" className="btn btn-primary" onClick={this.handleLogin}>Login</button>

</form>

</div>

:null

}

{

this.state.SigninDetails.length===0?

<div>

{console.log(this.state.SigninDetails.length)}

<h1> Login</h1>

<form>

<div className="form-group">

<label>Username (Preferrably Your E-Mail Id):</label>

<input style={{width:'40%'}} type="text" className="form-control" id="username" placeholder="Enter Name" name="username" onChange={this.allinone}/>

</div>

<div className="form-group">

<label>Password:</label>

<input style={{width:'40%'}} type="password" className="form-control" id="password" placeholder="Enter password" name="password" onChange={this.allinone} />

</div>

<button type="button" className="btn btn-primary" onClick={this.handleLogin}>Login</button>

</form>

</div>

:null

</div>

)

}

}

**src/Components/Question.js:**

import React, { Component } from 'react'

import axios from 'axios'

export default class Questions extends Component {

constructor(props) {

super(props)

this.state = {

username:props.username,

q1:'',

q2:'',

marksMessage:'',

errorMessage:''

}

this.allInOne=this.allInOne.bind(this)

this.handleSubmit=this.handleSubmit.bind(this)

}

allInOne(event)

{

this.setState({[event.target.name]:event.target.value})

}

handleSubmit(event)

{

console.log(this.state)

axios.post("http://localhost:5000/marks",this.state)

.then (res=>{

console.log(res.data)

this.setState({marksMessage:res.data})

})

.catch(error=>{

console.log("Data Loding Error")

this.setState({errorMessage:"Data Loding Error"})

})

}

render() {

return (

<div className="mt-3">

{

this.state.marksMessage.length?

<h1 className='text-success '>{this.state.marksMessage} </h1>

:

<div>

{this.props.username},Answer the following Questions

<form className="mt-5">

<div className="form-group ms-4">

<label> <h1>Q1 </h1> Which of the following is Acronym fro Full Stack Development?</label>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio1" name="q1" value="fsd" onChange={this.allInOne}/>

<label class="form-check-label" for="radio1">fsd</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio2" name="q1" value="agile" onChange={this.allInOne}/>

<label class="form-check-label" for="radio2">agile</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" name="q1" value="ml" onChange={this.allInOne}/>

<label class="form-check-label">ml</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" name="q1" value="tqm" onChange={this.allInOne}/>

<label class="form-check-label">tqm</label>

</div>

<br/>

</div>

<div className="form-group ms-4">

<label><h1>Q2</h1> Which of the following is Software Engineering course? </label>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio1" name="q2" value="fsd" onChange={this.allInOne}/>

<label class="form-check-label" for="radio1">fsd</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" id="radio2" name="q2" value="agile" onChange={this.allInOne}/>

<label class="form-check-label" for="radio2">agile</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" name="q2" value="ml" onChange={this.allInOne}/>

<label class="form-check-label">ml</label>

</div>

<div class="form-check">

<input type="radio" class="form-check-input" name="q2" value="tqm" onChange={this.allInOne}/>

<label class="form-check-label">tqm</label>

</div>

<br/>

<br/>

</div>

<button type="button" className="btn btn-primary" onClick={this.handleSubmit}>Submit Answers</button>

</form>

</div>

}

</div>

)

}

}

**src/Components/Signup.js:**  
import React from 'react';

import axios from 'axios'

class Signup extends React.Component {

constructor(props) {

super(props)

this.state = {

username:"",

password:"",

mobile:"",

SignupDetails:'',

error:''

}

this.allinone=this.allinone.bind(this);

this.handleSignup=this.handleSignup.bind(this)

}

allinone(event)

{

this.setState({[event.target.name]:event.target.value})

console.log(event.target.value)

}

handleSignup(event)

{

console.log(this.state)

axios.post("http://localhost:5000/signup",this.state)

.then (res=>{

console.log(res.data)

this.setState({SignupDetails:res.data})

})

.catch(error=>{

console.log("Data Loding Error")

this.setState({err:"Data Loding Error"})

})

}

render() {

return(

<div>

{

this.state.SignupDetails.length?

<div> <h1>{this.state.username}, You have Successfully Register with our system <br/>

Login into our system and take up assessment

</h1> </div>

: <div> <h1> Fill up the following details</h1>

<form>

<div className="form-group">

<label>Username (Preferrably Your E-Mail Id):</label>

<input style={{width:'40%'}} type="text" className="form-control" id="username" placeholder="Enter Name" name="username" onChange={this.allinone}/>

</div>

<div className="form-group">

<label>Password:</label>

<input style={{width:'40%'}} type="password" className="form-control" id="password" placeholder="Enter password" name="password" onChange={this.allinone} />

</div>

<div className="form-group">

<label>Mobile:</label>

<input style={{width:'40%'}} type="password" className="form-control" id="mobile" placeholder="Enter mobile" name="mobile" onChange={this.allinone} />

</div>

<button type="button" className="btn btn-primary" onClick={this.handleSignup}>Signup</button>

</form> </div>

}

</div>

)

}

}

export default Signup;

**/Backend/index.js:**

const express=require('express')

const bodyParser=require('body-parser');

var nodemailer = require('nodemailer');

const {MongoClient} = require('mongodb');

const app=express();

const cors=require('cors')

app.use(cors());

app.use(express.json())

app.use(bodyParser.urlencoded({

extended:true

}))

var transporter = nodemailer.createTransport({

service: 'Outlook365',

port:587,

auth: {

user: 'xxxx.21ads@sonatech.ac.in',

pass: xxxx

},

tls : { rejectUnauthorized: false }

});

app.get("/",function(req,res){

res.send("<h1> Welcome to node Js </h1>")

})

app.post("/signup",async function(req,res){

try {

let username=req.body.username;

let password=req.body.password;

let mobile=req.body.mobile;

console.log("hai");

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'Assessment';

await client.connect();

const db = client.db(database);

const myobj = { username: username, password:password,mobile:mobile };

const collection = db.collection('login');

const result = await collection.insertOne(myobj);

if (result.insertedCount === 0) {

throw new Error("Failed to insert data.");

}

console.log("Data inserted successfully");

res.send("<h1>Data Inserted successfully</h1>");

res.send("success")

// db.close();

} catch (error) {

console.error("Error:", error);

res.status(500).send("Failed to insert data.");

}

});

app.post("/signin", async function(req, res) {

try {

let username = req.body.username;

let password = req.body.password;

console.log("uuuuu" + username);

console.log("hai");

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'Assessment';

await client.connect();

const db = client.db(database);

const myobj = { username: username };

const collection = db.collection('login');

const result = await collection.find(myobj).toArray();

console.log("1");

if (result.length === 0) {

console.log("Username not found");

res.send("Failure"); // Username not found

return;

}

console.log(result[0].password + " Entered: " + password);

if (String(password) === String(result[0].password)) {

console.log("success")

res.send("success");

} else {

console.log("Failure");

res.send("Failure");

}

} catch (error) {

console.error("Error:", error);

res.status(500).send("Error occurred while signing in.");

}

});

app.post('/marks',async function(req,res){

try

{

let q1=req.body.q1;

let q2=req.body.q2;

let username=req.body.username;

let marks=0;

marks=(q1=='fsd')?marks+1:marks;

marks=(q2=="agile")?marks+1:marks;

console.log(marks)

marks=(marks/2) \* 100;

let dt=new Date();

console.log(dt.toString());

let dts=dt.toString();

const url = 'mongodb://localhost:27017';

const client = new MongoClient(url);

const database = 'Assessment';

await client.connect();

const db = client.db(database);

const myobj = {username:username,marks:marks,datetime:dts };

const collection = db.collection('marks');

const result = await collection.insertOne(myobj);

if (result.insertedCount === 0) {

throw new Error("Failed to insert data.");

}

console.log("Data inserted successfully");

// res.send("<h1>Marks Inserted successfully</h1>");

// res.send("success")

var mailOptions = {

from: 'ragavaa.21ads@sonatech.ac.in',

to: username,

subject: 'Sending your marks for the exam taken at Sona Assessment platform',

html: '<h1> Hai'+username +" You Secured Marks of "+ marks+" % in the exam taken on "+dt+"</h1>"

}

transporter.sendMail(mailOptions, function(error, info){

if (error) {

console.log(error);

} else {

console.log('Email sent: ' + info.response);

}

});

res.send("Hai "+username +" You Secured Marks of "+ marks+" % in the exam taken at "+dt +" and mail has been sent to you")

}

catch(error)

{

console.error("Error:", error);

//res.status(500).send("Data not found.");

}

});

app.listen(5000,function(){

console.log("Server is running on port number 5000")

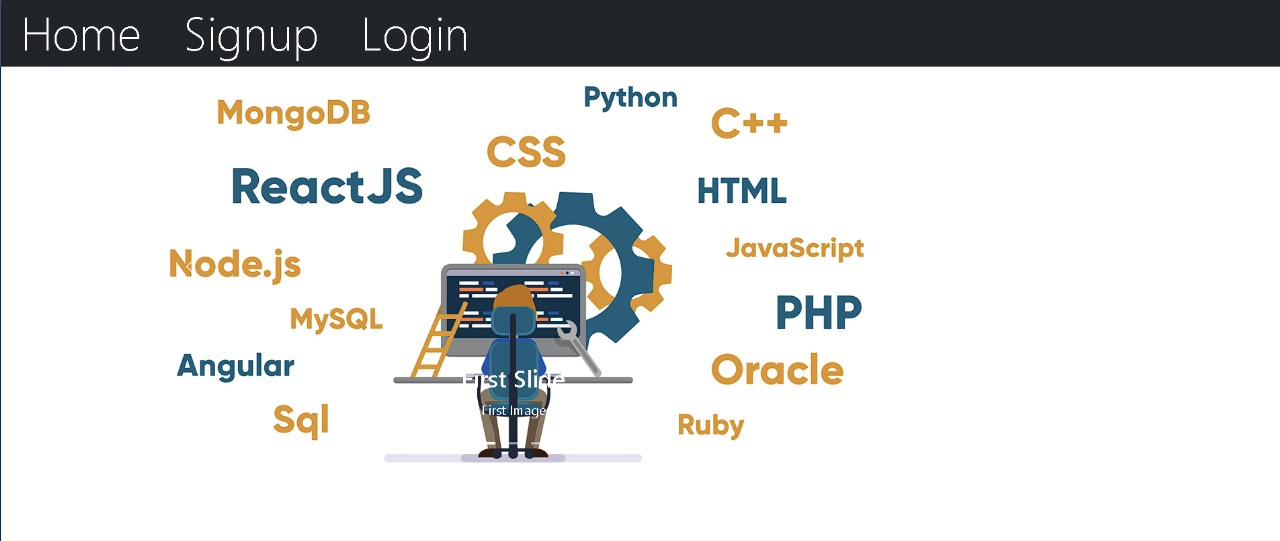
})

**OUTPUT:**

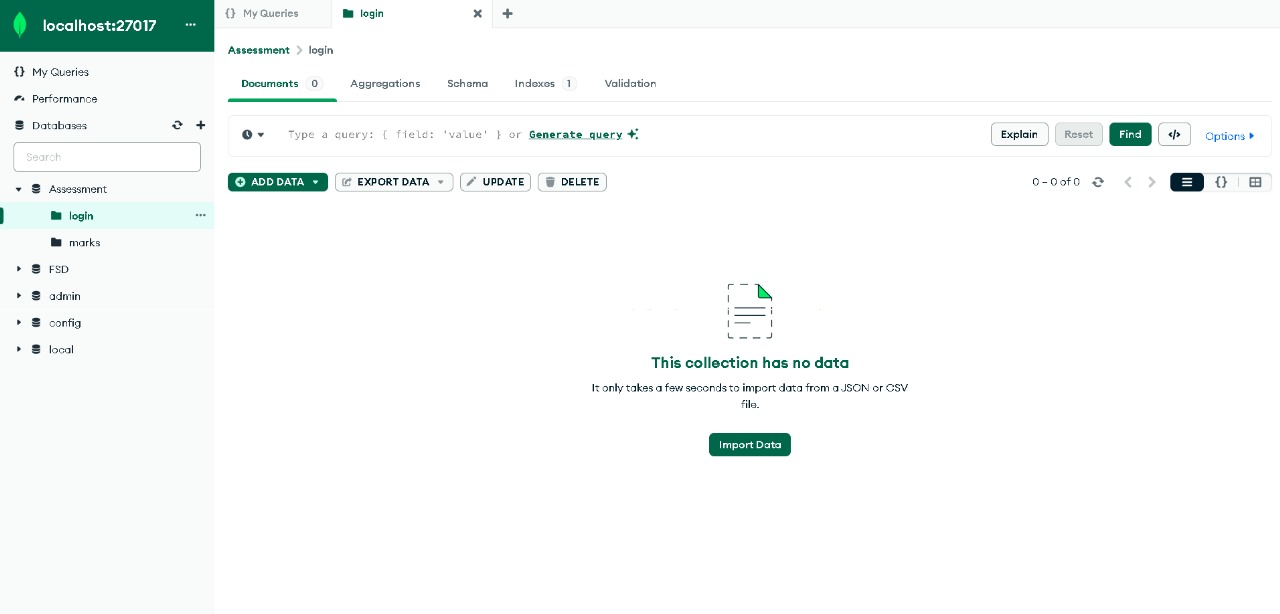




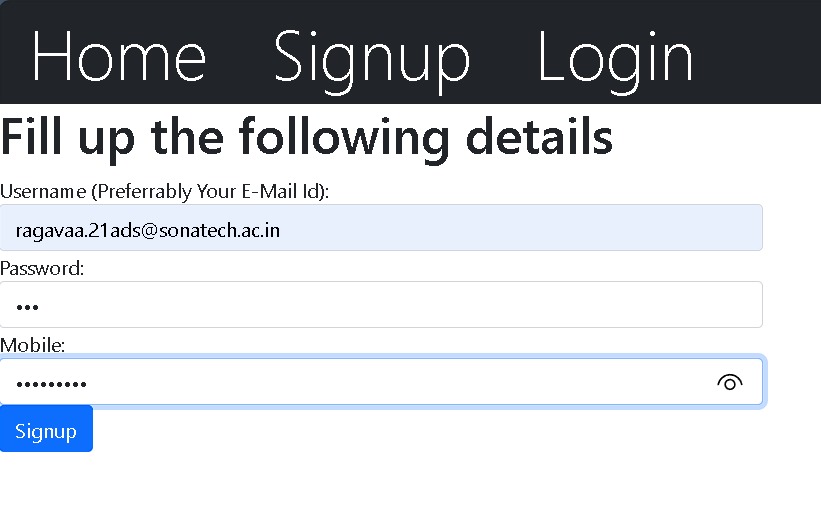
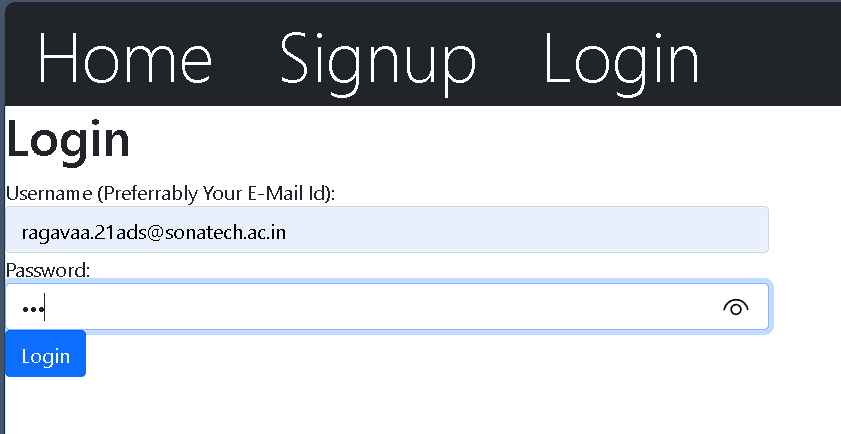
**HOME PAGE:**



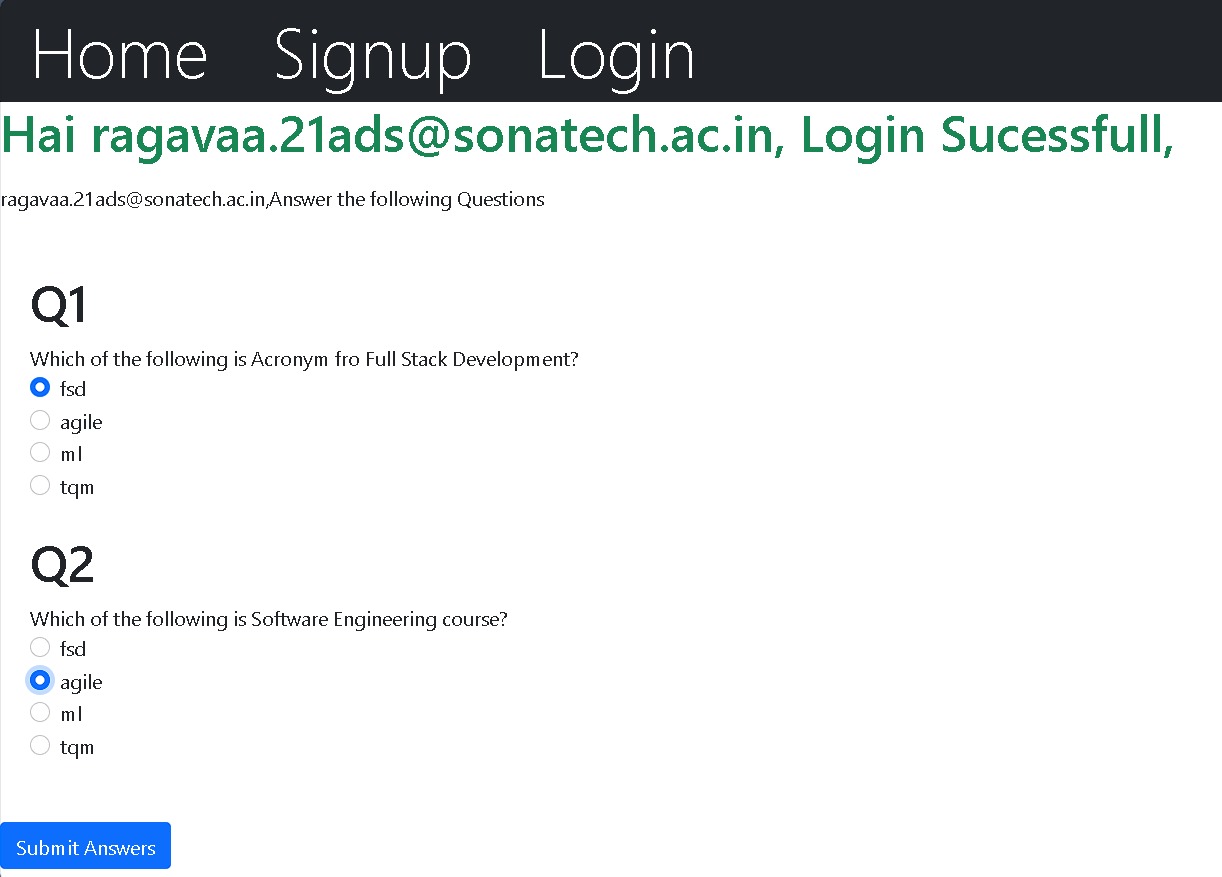
**MONGODB COMPASS:**

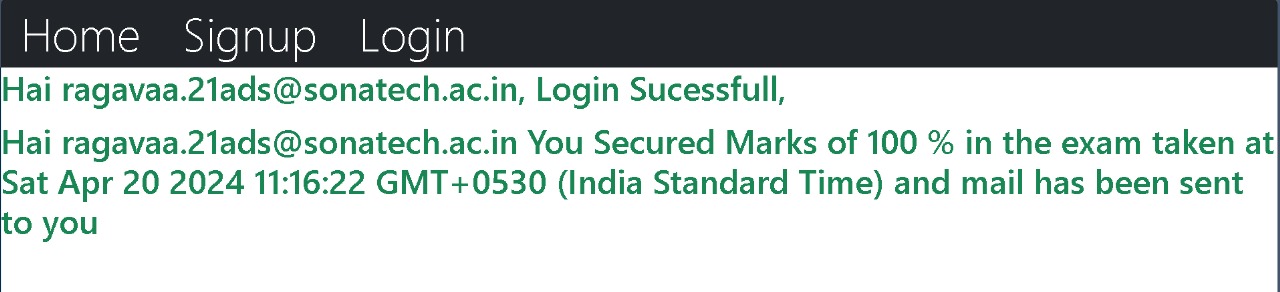


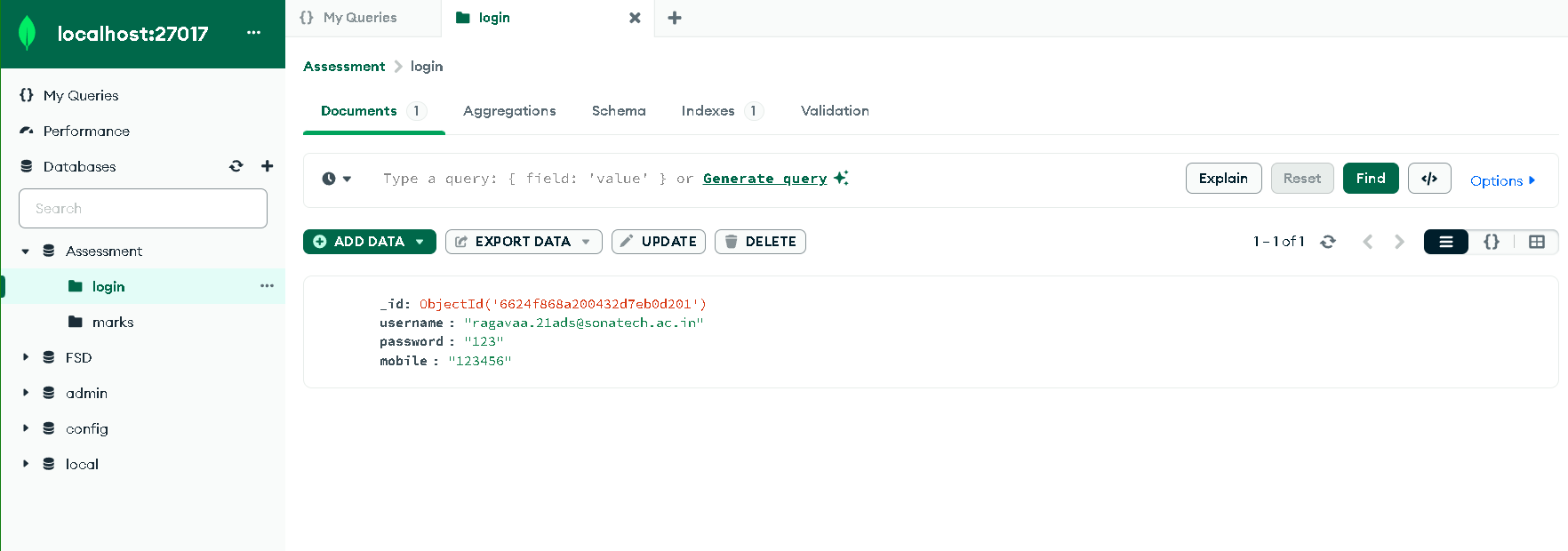
**SIGNUP:**

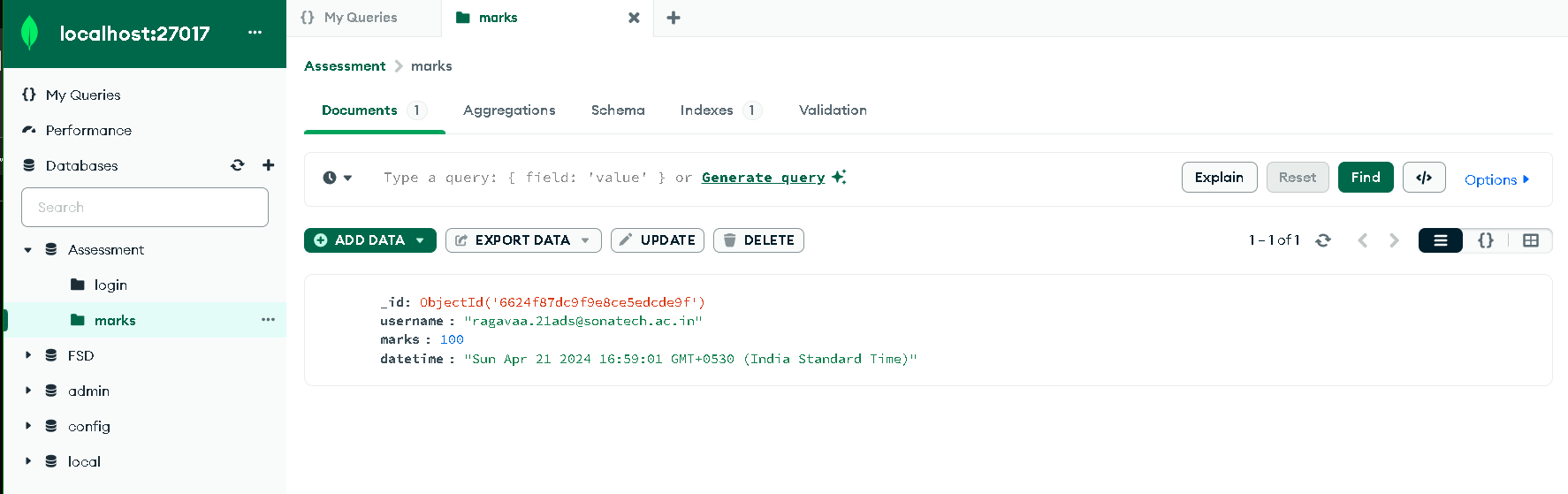
  
  
  
  
**LOGIN:**  
  


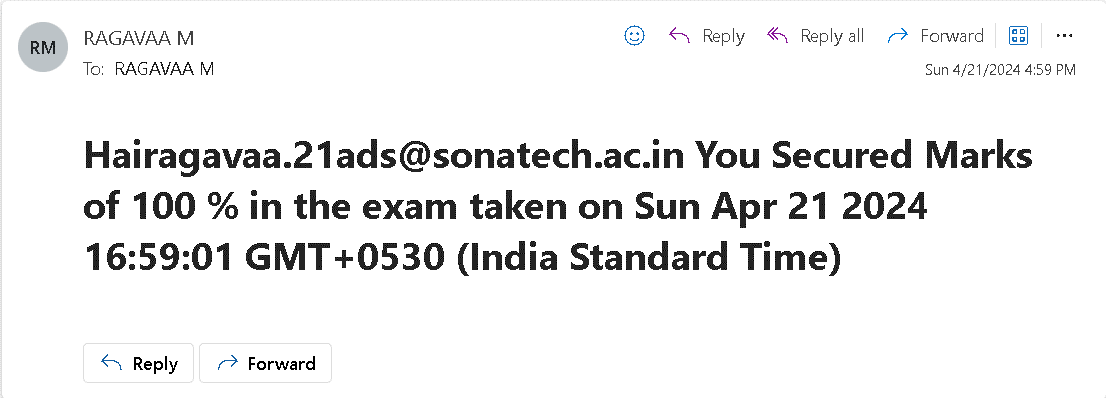
**ASSESSMENT AFTER SUCESSFUL LOGIN:**











**RESULT:**

Hence the program to execute the assessment using react js,mongodb and nodemailer is executed successfully.

|  |  |
| --- | --- |
| **EX NO: 10-(a)** | **BUILD AND DEPLOY NODE JS APPLICATION IN AWS BEANSTALK** |
| **DATE:22/04/2024** |

**AIM:**

Write a Node js program to give a customized greeting message to the client. Deploy

the Node js application AWS Beanstalk which offers Platform as a Service (PaaS) to the

clients.

**PROCEDURE:**

1. Create a free account in AWS cloud.

2. Create a directory called “Hello” and change to the directory. Give a following

command in command line environment to generate code skeleton for express based

node js application

* npx express-generator

3. Type the following command to install all dependency files

* npm install

4. To locally run the code, type the following command

* npm start

5. Login to AWS account (https://aws.amazon.com/console/)

6. Search service Text Box, type the following service “IAM” which is used for user

access management.

7. Click the link users. Create new user “lihan” by clicking add users button

8. Specify the user details as lihan-eb and click next.

9. Create a new user group by clicking create group

10 Give a group name as “lihan-eb” and search for the following policy

“AdministratorAccess-AWSElasticBeanstalk” and select it and create user group

11.Click next.

12. Review and Create. Click the create user.

13. New user lihan-eb is successfully created.

14. Click the user “lihan-eb”

15. Select the tab “Group” and add user to group.

16. Click the user “lihan-eb” and select the tab “Security Credentials” and click the

button “Create Access Key”

17. Store the generated access key and secret access key

Access key : AKIA25GBVHDHZX2O5DXZ

Secret access key: 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

18. Install the AWS cli for windows from the given link https://aws.amazon.com/cli/

19. Set the path variable to aws.exe (C:\Program Files\Amazon\AWSCLIV2)

20. Install the eb cli for windows using automated script (https://github.com/aws/aws

elastic-beanstalk-cli-setup).

21.Execute the following command

* python .\aws-elastic-beanstalk-cli-setup\scripts\ebcli\_installer.py

if fails due to non-availability of virtual environment then create a new virtual environment

* pip uninstall -y virtualenv
* pip install virtualenv

and to complete installation, ensure `eb` is in PATH.

22. Go to the project directory

aws configure --profile lihan-eb

AWS Access Key ID [None]: AKIA25GBVHDHZX2O5DXZ

AWS Secret Access Key [None]: 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

Default region name [None]:

Default output format [None]:

23. Execute the following command to create environment and deploy code in it

* E:\Hello>eb init --profile lihan-eb Select a default region and specify the location as ‘6’ for ap-south-1 : Asia Pacific (Mumbai).
* Select an application to use

1) blogify

2) express-app

3) [ Create new Application ]

(default is 3): 3

* Enter Application Name

(default is "Hello"):

Application Hello has been created.

It appears you are using Node.js. Is this correct?

(Y/n): Y

Select a platform branch.

1) Node.js 18 running on 64bit Amazon Linux 2

2) Node.js 16 running on 64bit Amazon Linux 2

3) Node.js 14 running on 64bit Amazon Linux 2

(default is 1): 2

* Do you want to set up SSH for your instances?

(Y/n): n

(.ebcli-virtual-env) E:\Hello>eb create

Enter Environment Name

(default is Hello-dev):

Enter DNS CNAME prefix

(default is Hello-dev):

* Select a load balancer type

1) classic

2) application

3) network

(default is 2): 2

* Would you like to enable Spot Fleet requests for this environment? (y/N): N

**PROGRAM:**

**routes/index.js:**

var express = require(‘express’);

var router = express.Router();

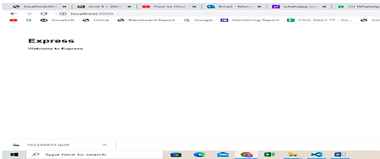
router.get(‘/’,function(req,res,next){

res.render(‘index’,{title : ‘Build and Deploy Node js Application using AWS Beanstalk}

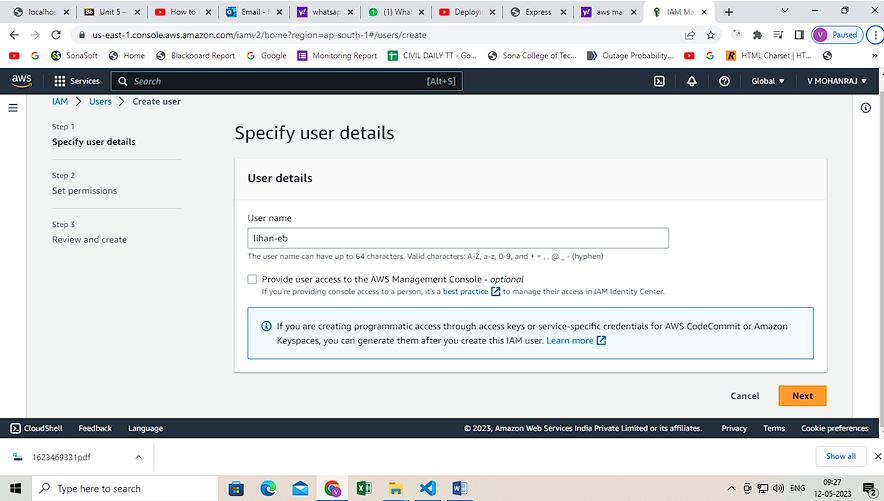
});

Module.exports = router;

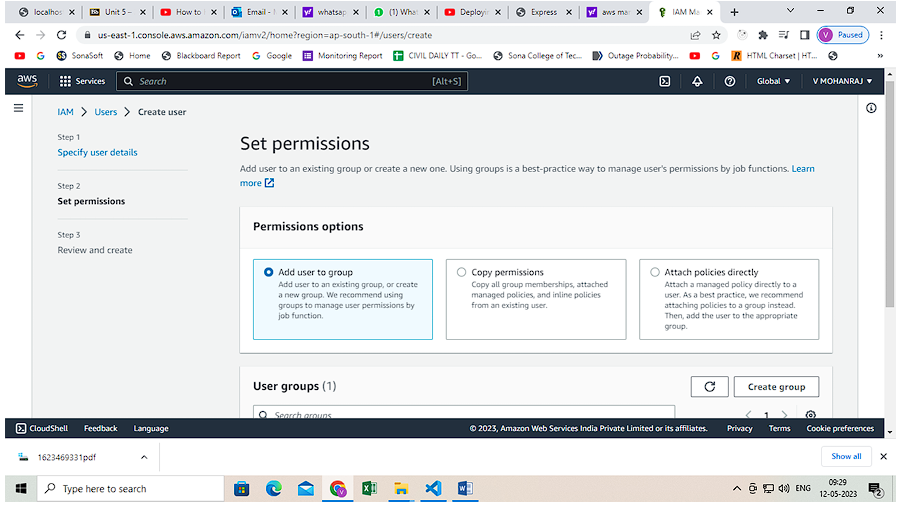
**OUTPUT:**

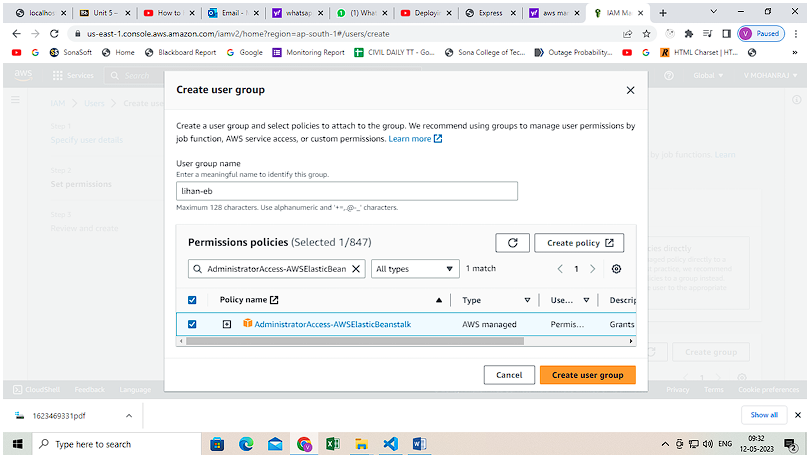
****

**USER DETAILS:**

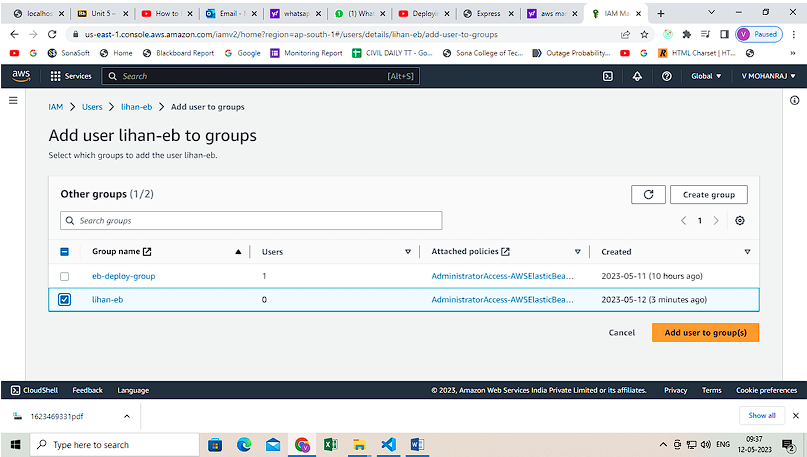
****

**CREATING NEW USER:**

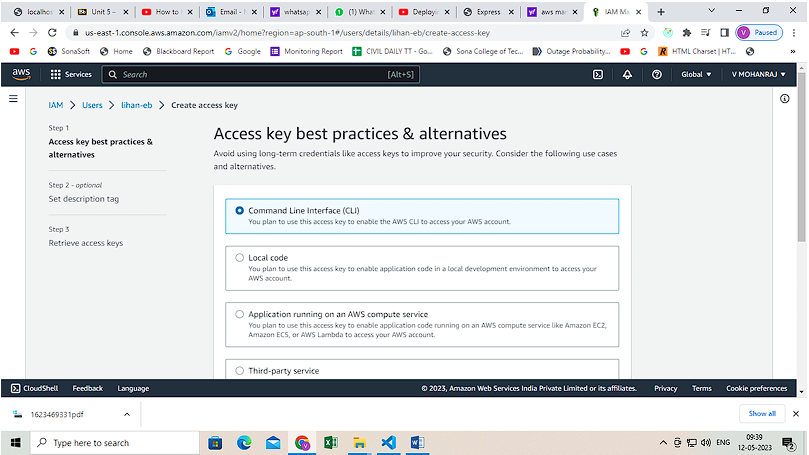
****

**GIVE GROUP NAME AND SELECT POLICY:  
  
  
**

**ADDING USER INSIDE THE GROUP:**

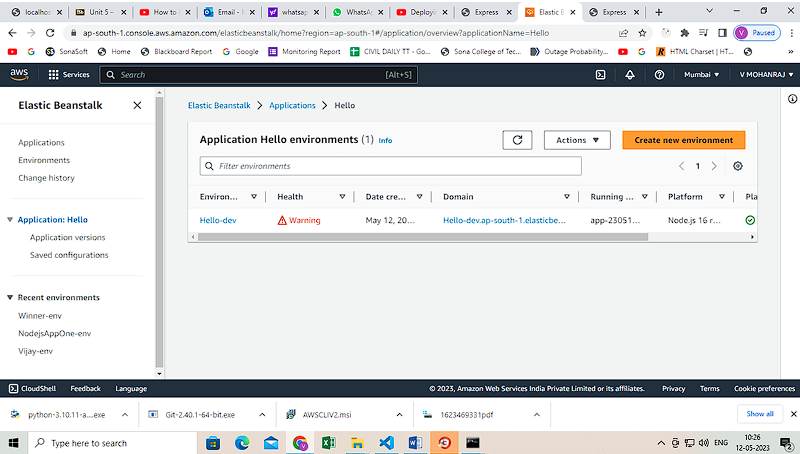
****

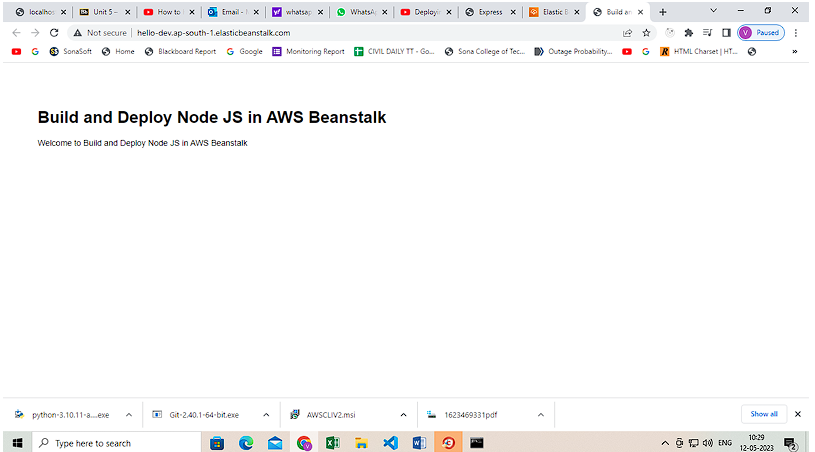
**SPECIFY ACCESS KEY:**

****

**After the successful deployment, go to your AWS login and Search for Elastic**

**Beanstalk and click the applications**

****

****

**RESULT:**

Hence the program to build and deploy an build and deploy node js application in aws beanstalk is executed successfully.

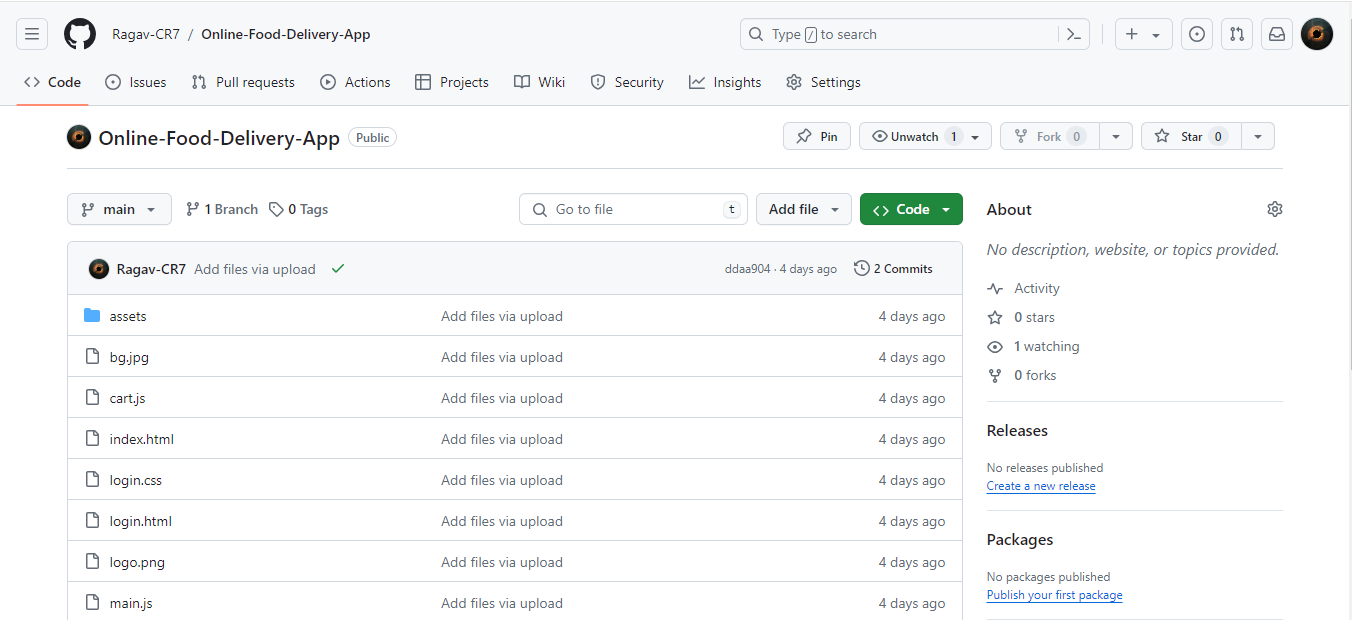
|  |  |
| --- | --- |
| **EX NO: 10-(b)** | **HOST AN WEBPAGE IN GITHUB** |
| **DATE:22/04/2024** |

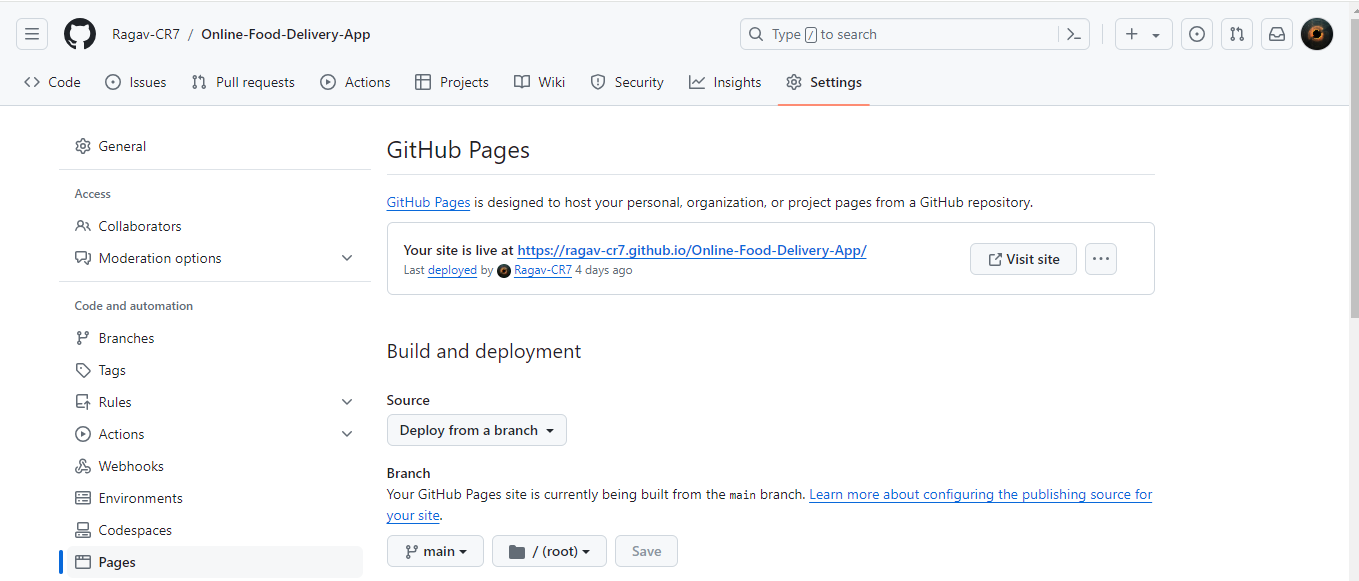
**AIM:**

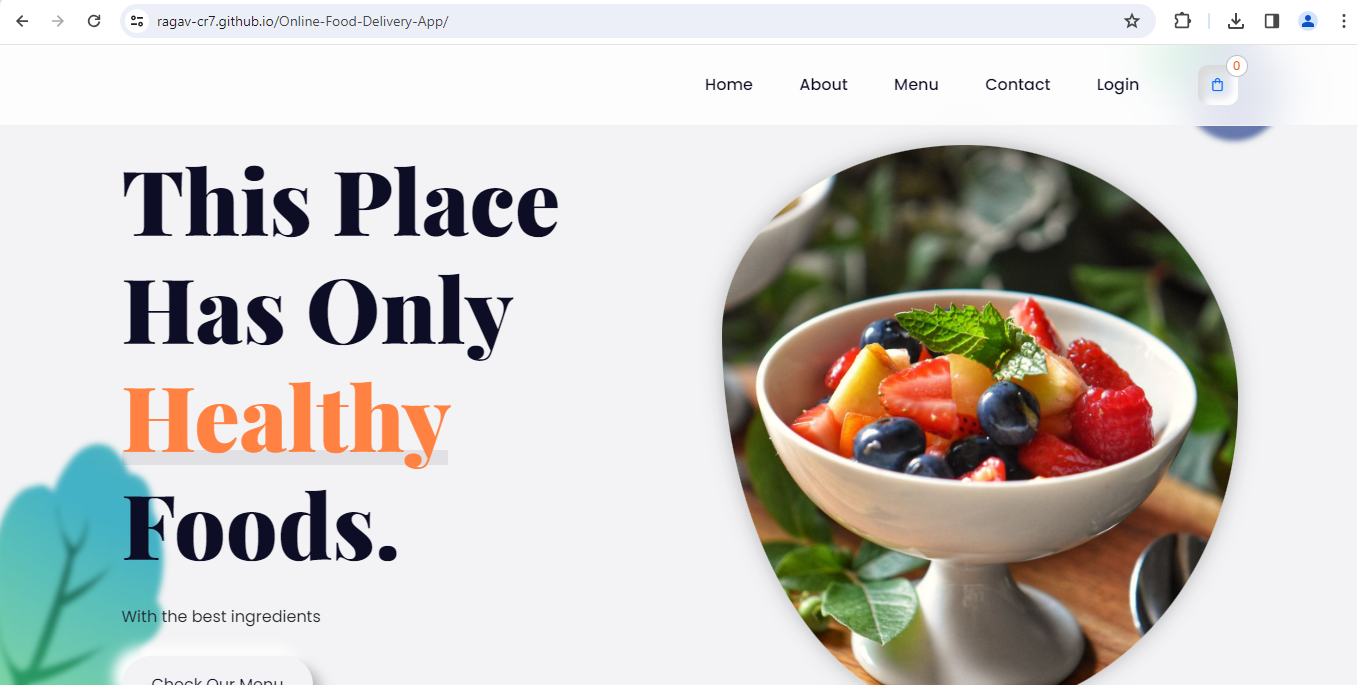
To host an webpage in github pages.  
  
**PROCEDURE:**

1. Create a GitHub Account:
2. If you don't already have one, sign up for a GitHub account at github.com.
3. Create a New Repository:
4. Click on the "+" icon in the top-right corner of the GitHub interface and select "New repository".
5. Give your repository a name. It's customary to name it something like "username.github.io", where "username" is your GitHub username. This naming convention is required for GitHub Pages to work.
6. Optionally, add a description for your repository.
7. Choose whether you want your repository to be public or private.
8. Click "Create repository".
9. Create Your Webpage:
10. You can either create a new HTML file directly in your repository or upload an existing HTML file.
11. Ensure that the main HTML file is named "index.html". This is the default file that GitHub Pages will look for when serving your webpage.
12. Commit Your Changes:
13. If you're creating a new HTML file, write your HTML code.
14. If you're uploading an existing HTML file, click on the "Add file" button and upload your HTML file.
15. Once your file(s) are added, scroll down to the "Commit changes" section.
16. Enter a commit message describing the changes you made.
17. Click on the "Commit changes" button.
18. Enable GitHub Pages:
19. Go to your repository's settings by clicking on the "Settings" tab.
20. Scroll down to the "GitHub Pages" section.
21. Under "Source", select "main" or "master" branch (or any other branch where your HTML file resides).
22. Click "Save".
23. Access Your Website:
24. After enabling GitHub Pages, GitHub will provide you with a URL where your website is hosted. It usually follows this pattern: https://username.github.io.
25. Your website should now be accessible at this URL.
26. Custom Domain (Optional):
27. If you have your own domain name, you can configure GitHub Pages to use it. This involves adding a CNAME file to your repository and configuring your domain registrar's DNS settings to point to GitHub's servers. .

**OUTPUT:**







**RESULT:**

Hence to host a webpage in github pages is executed successfully.

|  |  |
| --- | --- |
| **EX NO: 10-(c)** | **IMPLEMENT AXIOS IN REACT** |
| **DATE:22/04/2024** |

**AIM:**

To implement axios library in react js.  
  
**PROCEDURE:**

1. Adding Axios to the Project.
2. Making a GET Request.
3. Making a POST Request.
4. Making a DELETE Request.
5. Using a Base Instance in Axios.
6. Using async and await.

**PROGRAM:**

**/src/components/PersonList.js:**

import React from 'react';

import axios from 'axios';

export default class PersonList extends React.Component {

state = {

persons: []

}

componentDidMount() {

axios.get(`https://jsonplaceholder.typicode.com/users`)

.then(res => {

const persons = res.data;

this.setState({ persons });

})

}

render() {

return (

<ul>

{

this.state.persons

.map(person =>

<li key={person.id}>{person.name}</li>

)

}

</ul>

)

}

}

**/src/components/PersonAdd.js:**

import React from 'react';

import axios from 'axios';

export default class PersonAdd extends React.Component {

state = {

name: ''

}

handleChange = event => {

this.setState({ name: event.target.value });

}

handleSubmit = event => {

event.preventDefault();

const user = {

name: this.state.name

};

axios.post(`https://jsonplaceholder.typicode.com/users`, { user })

.then(res => {

console.log(res);

console.log(res.data);

})

}

render() {

return (

<div>

<form onSubmit={this.handleSubmit}>

<label>

Person Name:

<input type="text" name="name" onChange={this.handleChange} />

</label>

<button type="submit">Add</button>

</form>

</div>

)

}}

**/src/components/PersonRemove.js:**

import React from 'react';

import API from '../api';

export default class PersonRemove extends React.Component {

state = {

id: ''

}

handleChange = event => {

this.setState({ id: event.target.value });

}

handleSubmit = async(event) => {

event.preventDefault();

const response = await API.delete(`users/${this.state.id}`);

console.log(response);

console.log(response.data);

}

render() {

return (

<div>

<form onSubmit={this.handleSubmit}>

<label>

Person ID:

<input type="number" name="id" onChange={this.handleChange} />

</label>

<button type="submit">Delete</button>

</form>

</div>

)

}

}

**src/api.js:**

import axios from 'axios';

export default axios.create({

baseURL: `http://jsonplaceholder.typicode.com/`

});

**src/App.js:**

import PersonList from './components/PersonList';

import PersonAdd from './components/PersonAdd';

import PersonRemove from './components/PersonRemove';

function App() {

return (

<div ClassName="App">

<PersonList/>

<PersonAdd/>

<PersonRemove/>

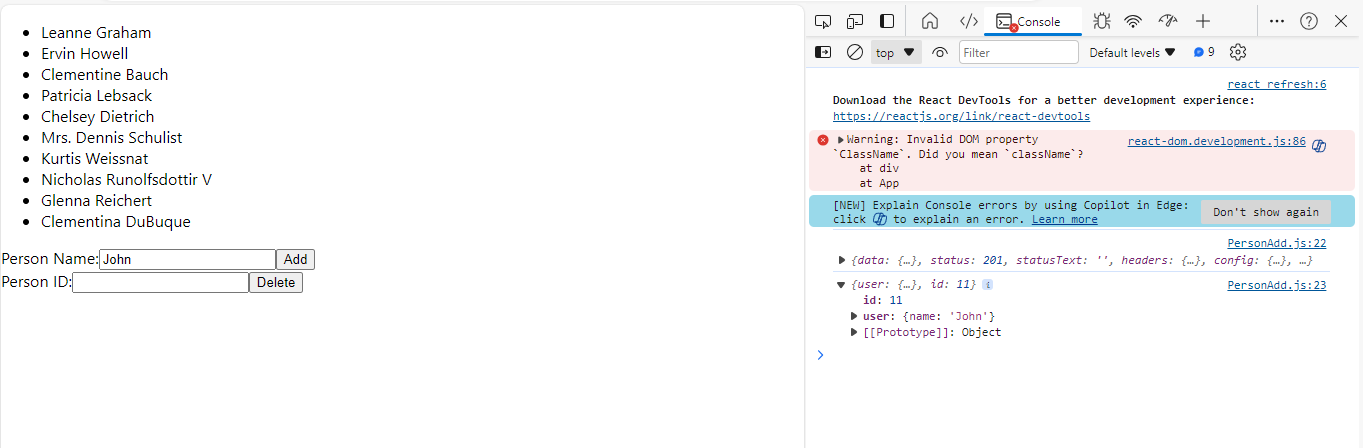
</div>

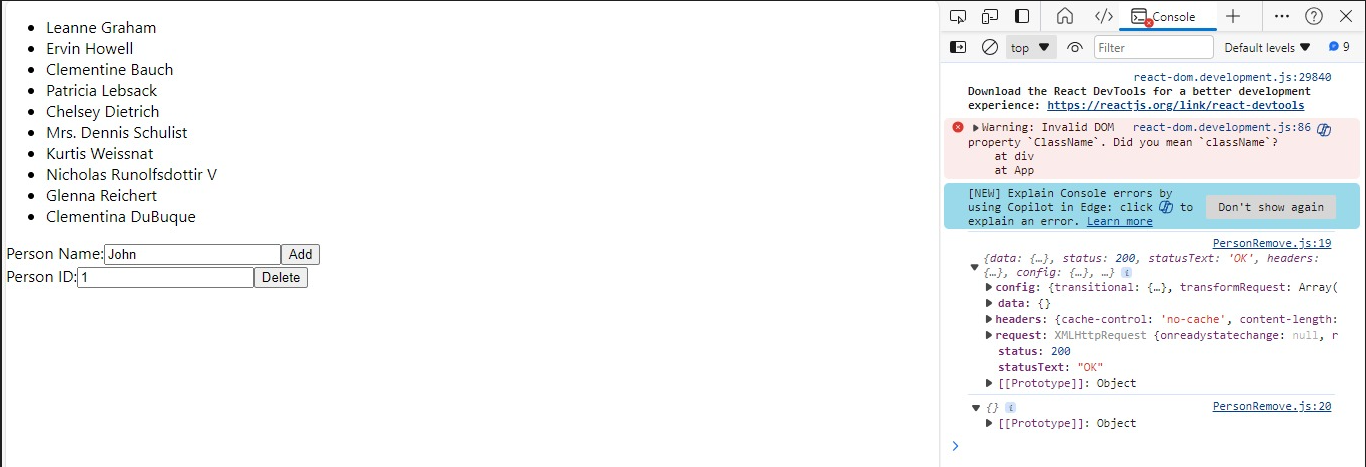
)

}

export default App;

**OUTPUT:**





**RESULT:**

Hence the program to implement axios library in react is executed successfully.