|  |  |  |
| --- | --- | --- |
| **Name: Abhinav Sankarshana Dasu** | **SRN:PES2UG22CS017** | **Section: A** |
| **Date:22/11/23** | **Unit 4 Assignment Exercise** |
| **PROBLEM STATEMENT1 (for odd number SRNs):** | | |
| Create a Node.js application that interacts with a MongoDB database named Bangalore\_City. This database contains a collection of restaurants, and each restaurant document should include the following entities:  rest\_name: Name of the restaurant. rest\_id: Unique identifier for the restaurant. rest\_addr: Address of the restaurant.  rest\_reviews: Grade assigned to the restaurant. Food\_menu: Menu items offered by the restaurant.  Implement CRUD operations for the "restaurants" collection:   1. Create a new restaurant:   Design a route that allows the addition of a new restaurant to the collection. Users should be able to provide details such as rest\_name, rest\_addr, rest\_grade, rest\_reviews, and Food\_menu.   1. Read all restaurants:   Create a route that retrieves and displays all restaurants in the collection.   1. Update a restaurant's grade:   Design a route that allows users to update the rest\_grade of a specific restaurant based on the rest\_id. | | |

|  |
| --- |
| **OBJECTIVE** |
| The objective of this exercise is to test the student on back end frame work and storage Node JS with Mongo DB. It evaluates the student’s knowledge Node Js App, modules, Node Js ,HTTP modules, Reading and writing to Mongo DB  through Node Js. |
| **PREREQUISITE** |
| In order to complete this exercise, the student needs to understand the fundamentals of JavaScript, Mongo DB Operations with Nodejs modules. |
| **PROGRAM**  App.js |

import React, { useState, useEffect } from 'react';

import axios from 'axios';

function App() {

const [restaurants, setRestaurants] = useState([]);

const [newRestaurant, setNewRestaurant] = useState({

    rest\_name: '',

    rest\_id: '',

    rest\_addr: '',

    rest\_reviews: 0,

    food\_menu: [],

});

const [updateGrade, setUpdateGrade] = useState({

    rest\_id: '',

    rest\_reviews: 0,

});

const [error, setError] = useState(null);

useEffect(() => {

axios.get('http://localhost:3000/restaurants')

.then(response => setRestaurants(response.data))

.catch(error => console.error('Error fetching restaurants:', error));

}, []);

const handleAddRestaurant = () => {

    axios.post('http://localhost:3000/restaurants',

    newRestaurant)

    .then(response => {

    setRestaurants([...restaurants, response.data]);

    setNewRestaurant({

    rest\_name: '',

    rest\_id: '',

    rest\_addr: '',

    rest\_reviews: 0,

    food\_menu: [],

    });

})

.catch(error => {

    console.error('Error adding restaurant:', error);

    if (error.response) {

        console.error('Server responded with status:',

        error.response.status);

        console.error('Response data:', error.response.data);

        setError(`Server responded with status

        ${error.response.status}. Please check the server logs.`);

    }

    else if (error.request) {

        console.error('No response received from the server.');

        setError('No response received from the server.Please check the server logs.');

    }

    else {

        console.error('Error setting up the request:',error.message);

        setError('Error setting up the request. Please check the server logs.');

    }

});

};

const handleUpdateGrade = () => {

    axios.put(`http://localhost:3000/restaurants/${updateGrade.rest\_id}`, { rest\_reviews: updateGrade.rest\_reviews })

    .then(response => {setRestaurants(restaurants.map(restaurant =>

        (restaurant.rest\_id === updateGrade.rest\_id ? response.data :restaurant)));

        setUpdateGrade({

            rest\_id: '',

            rest\_reviews: 0,

    });

})

.catch(error => {

    console.error('Error updating grade:', error);

    if (error.response) {

        console.error('Server responded with status:',

        error.response.status);

        console.error('Response data:', error.response.data);

        setError(`Server responded with status

        ${error.response.status}. Please check the server logs.`);

    }

    else if (error.request) {

        console.error('No response received from the server.');

        setError('No response received from the server.Please check the server logs.');

    }

    else {

        console.error('Error setting up the request:',

        error.message);

        setError('Error setting up the request. Please check the server logs.');

    }

});

};

return(

<div>

    <h1>Restaurant App</h1>

    <h2>Restaurants</h2>

    <ul>

        {restaurants.map(restaurant => (

        <li key={restaurant.rest\_id}>

            {restaurant.rest\_name} - Grade:

            {restaurant.rest\_reviews}

            </li>

            ))}

            </ul>

            <h2>Add New Restaurant</h2>

            <div>

                <label>Name:</label>

                <input type="text" value={newRestaurant.rest\_name}

                onChange={e => setNewRestaurant({ ...newRestaurant, rest\_name:e.target.value })} />

                </div>

                <div>

                    <label>ID:</label>

                    <input type="text" value={newRestaurant.rest\_id}onChange={e => setNewRestaurant({ ...newRestaurant, rest\_id:e.target.value })} />

                    </div>

                    <div>

                        <label>Address:</label>

                        <input type="text" value={newRestaurant.rest\_addr}onChange={e => setNewRestaurant({ ...newRestaurant, rest\_addr:e.target.value })} />

                        </div>

                        <div>

                            <label>Grade:</label>

                            <input type="number" value={newRestaurant.rest\_reviews}onChange={e => setNewRestaurant({ ...newRestaurant,rest\_reviews: e.target.value })} />

                            </div>

                            <div>

                                <label>Food Menu:</label>

                                <input type="text" value={newRestaurant.food\_menu}onChange={e => setNewRestaurant({ ...newRestaurant, food\_menu:e.target.value.split(',') })} />

                                </div>

                                <button onClick={handleAddRestaurant}>Add Restaurant</button>

                                <h2>Update Restaurant Grade</h2>

                                <div>

                                    <label>ID:</label>

                                    <input type="text" value={updateGrade.rest\_id} onChange={e=> setUpdateGrade({ ...updateGrade, rest\_id: e.target.value })}/>

                                    </div>

                                    <div>

                                        <label>New Grade:</label><input type="number" value={updateGrade.rest\_reviews}onChange={e => setUpdateGrade({ ...updateGrade, rest\_reviews:e.target.value })} />

                                        </div>

                                        <button onClick={handleUpdateGrade}>Update Grade</button>

                                        {error && <p style={{ color: 'red' }}>{error}</p>}

                                        </div>

    );

}

export default App;

**Server.js**

const express = require("express");

const MongoClient = require("mongodb").MongoClient;

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

const app = express();

const Url = "mongodb://127.0.0.1:27017";

const dbname = "Bangalore\_city";

let db;

const client = new MongoClient(Url);

client.connect()

.then(client => {

    db = client.db(dbname);

    createDatabase(client);

})

.catch(error => {

    console.log("Error connecting to MongoDB:", error.message);

    process.exit(1);

});

async function createDatabase(client){

    try{

        const adminDB = client.db("admin");

        const databases = await adminDB.admin().listDatabases();

        if(!databases.databases.some(dbInfo=>dbInfo.name===dbname)){

            await adminDB.admin().command({create:dbname});

        }

    }

    catch(error){

        console.log("Error creating database:",error.message);

        process.exit(1);

    }

}

app.use(bodyParser.json());

app.use((req, res, next) => {

    res.header('Access-Control-Allow-Origin', '\*');

    res.header('Access-Control-Allow-Headers', 'Origin,X-Requested-With, Content-Type, Accept');

    next();

    });

    app.post('/restaurants', async (req, res) => {

        const newRestaurant = req.body;

        try {

          const result = await db.collection('restaurants').insertOne(newRestaurant);

          res.json(result);

        } catch (error) {

          console.error('Error creating a new restaurant:', error);

          res.status(500).json({ error: 'An error occurred while creating the restaurant' });

        }

      });

    app.get('/restaurants', async (req, res) => {

    try {

    const restaurants = await

    db.collection('restaurants').find().toArray();

    res.json(restaurants);

    } catch (error) {

    console.error('Error reading all restaurants:',

    error.message);

    res.status(500).json({ error: 'Internal Server Error'});}

});

app.put('/restaurants/:rest\_id', async (req, res) => {

try {

const { rest\_id } = req.params;

const { rest\_reviews } = req.body;

const result = await

db.collection('restaurants').findOneAndUpdate(

{ rest\_id },

{ $set: { rest\_reviews } },

{ returnDocument: 'after' }

);

res.json(result.value);

} catch (error) {

console.error('Error updating a restaurant\'s grade:',

error.message);

res.status(500).json({ error: 'Internal Server Error' });

}

});

/\*app.get('/', (req, res) => {

res.send('Welcome to the Bangalore City Restaurants API');

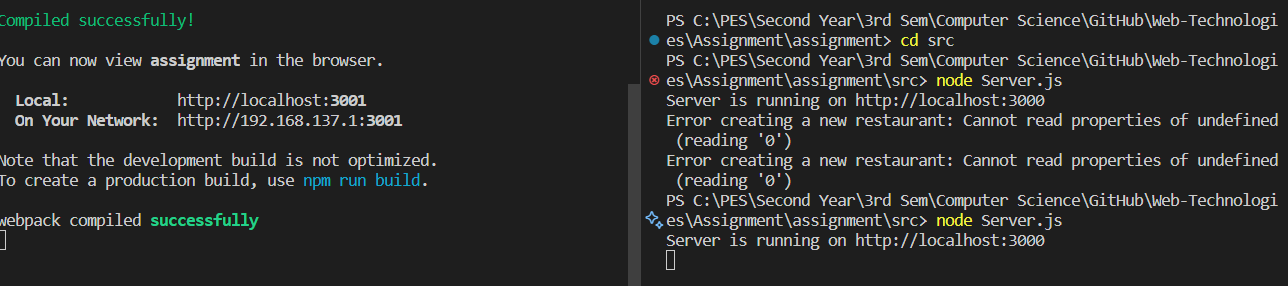
});\*/

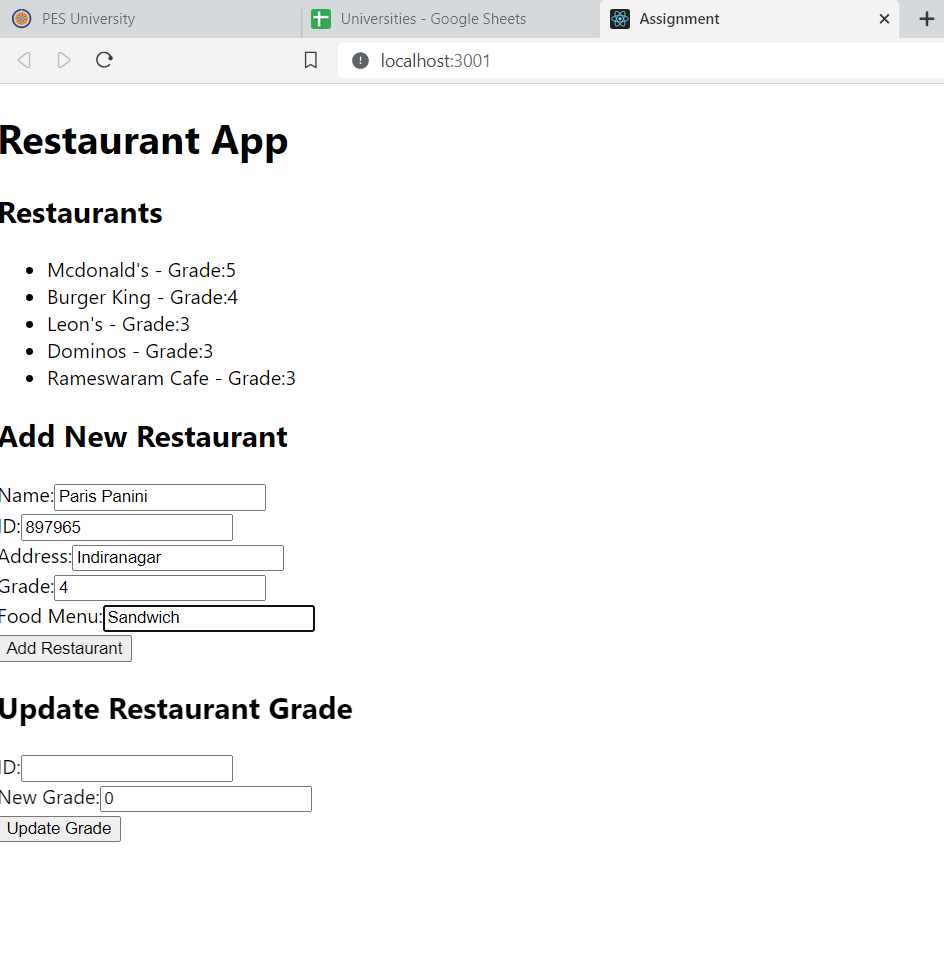
app.listen(3000,() => {

console.log(`Server is running on http://localhost:3000`);

});

# SCREENSHOT OF YOUR OUTPUT







**Unit 4: NODE JS & Mongo DB**

**2022**

