Name: ANKITH GOWDA B S	SRN:PES2UG22CS077	Section: B
	Date:23/11/23	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.

2. Read all restaurants:
   Create a route that retrieves and displays all restaurants in the collection.

3. 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

```
// src/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'); const bodyParser =
require('body-parser');

const app = express(); const PORT = 3000;

const mongoUrl =

'mongodb://localhost:27017'; const dbName

=

'Bangalore_City'; let db;
```

```
MongoClient.connect(mongoUrl, { useNewUrlParser: true,
useUnifiedTopology: true })
  .then((client) => { db =
     client.db(dbName);
     createDatabase(client)
     ;
  })
  .catch((error) => { console.error('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.error('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) => { try
  { const newRestaurant = req.body; const result
= await
```

```
db.collection('restaurants').insertOne(newRestaurant);
res.json(result.ops[0]);
  } catch (error) { console.error('Error creating a
     new restaurant:',
error.message); res.status(500).json({ error: 'Internal Server
Error' }); } });

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(PORT, () => { console.log(`Server is running on
http://localhost:${PORT}`); });
```

SCREENSHOT OF YOUR OUTPUT

Add New Restaurant

Name: hello
ID: 1
Address: brigade road
Grade: 3
Food Menu: pizza, bread
Add Restaurant

Restaurants

- hello - Grade: 3
- hello 1 - Grade: 5

Update Restaurant Grade

ID:
New Grade: 0
Update Grade


```
_id: ObjectId('655c47d04b75185b13c1b9d9')
rest_name: "hello"
rest_id: "1"
rest_addr: "brigade road"
rest_reviews: "3"
▾ food_menu: Array (2)
    0: "pizza"
    1: " bread"
```