Ganpat University | Institute of Computer Technology

॥ विद्यया समाजोत्कर्षः ॥

Name: Tushar Panchal

En.No: 21162101014

Sub: MICROSEVICES

Branch: CBA

Batch:51

----------PRACTICAL 08----------

❖ Question (TASK) :

Implement an application with CRUD operation using MongoDB withNodeJS.
Sympoms Pvt Ltd company wants to manage their employee records and this task is given to you. They are seeking functionalities:

Practical 8.1: *New joiner data should be stored in DB.*
Practical 8.2: *The admin can see the whole DB,*
Practical 8.3: *The admin can filter out the DB.*
Practical 8.4: *The admin can delete the record from DB.*
Practical 8.5: *Create a database with the name "student". Create a collection named "cba". Perform the following tasks:*

→Insert the details of 10 students with the following details: Name, age, address, phone number, and email.

→Filter the students whose age is greater than 16.

→Delete the student record whose phone number contains the digit '6' in it.

Github Link :
https://github.com/Tushar007079/MICROSERVICES_PRACTICALS/tree/0f32ac8c57547d15e67ef1da056f94f80cf24397/8

STEPS TO PERFORM THIS TASK :

Step 1 :

- Install Dependencies.
- Open a terminal or command prompt, navigate to the project directory, and install the required dependencies. In this case, we need **'cli-table','mongoose'** to run the server.
- Run this following command to initialize the package.json file :
  `npm init -y`
- Run this following command to install the mongoose , cli-table :

```
npm install mongoose cli-table
```

Step 2 :

- Create your MongoDB Database and collection.
- In the MongoDB shell, create a database named "student" and a collection named "cba" using the following commands:
  `use student`
  `Db.createCollection("cba")`

Step 3 :

- **Create Separate JavaScript Files:**
- Create separate JavaScript files for each practical task (8_1.js, 8_2.js, 8_3.js, 8_4.js, 8_5.js).

Step 4 :

- Run the server.
- In the Terminal/CMD , run this following command to start the NodeJS Server:

```
node app.js
```

✓ **8_1.js :-**

```
const mongoose = require('mongoose');
const readline = require('readline');
const Table = require('cli-table3');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true,
useUnifiedTopology: true });

const Student = mongoose.model('Student', {
    name: String,
    age: Number,
    address: String,
    phoneNumber: String,
    email: String
});

// Create a function to get user input for the number of students to insert
function getNumberOfStudents() {
    rl.question('Enter the number of students you want to insert: ',
(numStudentsInput) => {
        const numStudents = parseInt(numStudentsInput);
```

```
        insertStudents(numStudents);
    });
}

// Create a function to get user input for student details and insert students
function insertStudents(numStudents) {
    const studentsData = [];

    function insertStudent(index) {
        if (index < numStudents) {
            rl.question(`Enter details for Student ${index + 1}:\nName: `,
(name) => {
                rl.question('Age: ', (age) => {
                    rl.question('Address: ', (address) => {
                        rl.question('Phone Number: ', (phoneNumber) => {
                            rl.question('Email: ', (email) => {
                                studentsData.push({
                                    name: name,
                                    age: parseInt(age),
                                    address: address,
                                    phoneNumber: phoneNumber,
                                    email: email
                                });

                                insertStudent(index + 1); // Insert the next
student
                            });
                        });
                    });
                });
            });
        } else {
            // All students are collected, insert them into the database
            Student.insertMany(studentsData)
                .then((students) => {
                    console.log(`\nInserted ${students.length} students:`);
                    displayTable(students);

                    rl.close(); // Close the readline interface
                    mongoose.disconnect(); // Close the database connection
                })
                .catch((err) => {
                    console.error(err);
                    rl.close(); // Close the readline interface
                    mongoose.disconnect(); // Close the database connection
                });
        }
    }
```

```
    insertStudent(0); // Start inserting students
}

// Function to display data in a table format
function displayTable(data) {
    const table = new Table({
        head: ['Name', 'Age', 'Address', 'Phone Number', 'Email'],
        colWidths: [20, 6, 30, 15, 30],
    });

    data.forEach((item) => {
        table.push([item.name, item.age, item.address, item.phoneNumber,
item.email]);
    });

    console.log(table.toString());
}

// Start by getting the number of students to insert
getNumberOfStudents();
```

- ✓ **Output :**

```
pwsh  811  119ms                                  20.4.0  61  10,11:36
node 1.js
Enter the number of students you want to insert: 2
Enter details for Student 1:
Name: James Bond
Age: 20
Address: New York
Phone Number: 007
Email: jamesbond@gmail.com
Enter details for Student 2:
Name: Tom Cruse
Age: 15
Address: NYC
Phone Number: 11116
Email: tom@mail.com

Inserted 2 students:
```

Name	Age	Address	Phone Number	Email
James Bond	20	New York	007	jamesbond@gmail.com
Tom Cruse	15	NYC	11116	tom@mail.com

- ✓ **8_2.js :-**

```
const mongoose = require('mongoose');
const Table = require('cli-table3');

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true,
useUnifiedTopology: true });

const Student = mongoose.model('Student', {
    name: String,
    age: Number,
```

```
    address: String,
    phoneNumber: String,
    email: String
});

// Find all students and display them in a table format
Student.find({})
    .then((students) => {
        console.log(`\nAll Students:`);
        displayTable(students);
    })
    .catch((err) => {
        console.error(err);
    })
    .finally(() => {
        mongoose.disconnect(); // Close the database connection
    });

// Function to display data in a table format
function displayTable(data) {
    const table = new Table({
        head: ['Name', 'Age', 'Address', 'Phone Number', 'Email'],
        colWidths: [20, 6, 30, 15, 30],
    });

    data.forEach((item) => {
        table.push([item.name, item.age, item.address, item.phoneNumber,
item.email]);
    });

    console.log(table.toString());
}
```

✓ **Output :**

```
pwsh  811  1s 257ms                                  20.4.0  79  10,11:56
node 2.js

All Students:
```

Name	Age	Address	Phone Number	Email
James Bond	20	New York	007	jamesbond@gmail.com
Tom Cruse	15	NYC	11116	tom@mail.com

✓ **8_3.js :-**

```
const mongoose = require('mongoose');
const readline = require('readline');
const Table = require('cli-table3');

const rl = readline.createInterface({
```

```
    input: process.stdin,
    output: process.stdout
});

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true,
useUnifiedTopology: true });

const Student = mongoose.model('Student', {
    name: String,
    age: Number,
    address: String,
    phoneNumber: String,
    email: String
});

// Create a function to get user input for filtering students by age
function getAgeFilter() {
    rl.question('Enter the minimum age to filter students: ', (minAgeInput) =>
{
        const minAge = parseInt(minAgeInput);
        filterStudentsByAge(minAge);
    });
}

// Function to filter students by age and display them in a table format
function filterStudentsByAge(minAge) {
    Student.find({ age: { $gt: minAge } })
        .then((filteredStudents) => {
            console.log(`\nStudents older than ${minAge} years:`);
            displayTable(filteredStudents);
        })
        .catch((err) => {
            console.error(err);
        })
        .finally(() => {
            rl.close(); // Close the readline interface
            mongoose.disconnect(); // Close the database connection
        });
}

// Function to display data in a table format
function displayTable(data) {
    const table = new Table({
        head: ['Name', 'Age', 'Address', 'Phone Number', 'Email'],
        colWidths: [20, 6, 30, 15, 30],
    });

    data.forEach((item) => {
```

```
        table.push([item.name, item.age, item.address, item.phoneNumber,
item.email]);
    });

    console.log(table.toString());
}

// Start by getting the minimum age to filter students
getAgeFilter();
```

✓ **Output :**

```
pwsh  811  443ms                                  20.4.0   79   10,11:56
node 3.js
Enter the minimum age to filter students: 12

Students older than 12 years:
```

Name	Age	Address	Phone Number	Email
James Bond	20	New York	007	jamesbond@gmail.com
Tom Cruse	15	NYC	11116	tom@mail.com

```
pwsh  811  10s 230ms                              20.4.0   79   10,12:00
node 3.js
Enter the minimum age to filter students: 15

Students older than 15 years:
```

Name	Age	Address	Phone Number	Email
James Bond	20	New York	007	jamesbond@gmail.com

✓ **8_4.js :-**

```
const mongoose = require('mongoose');
const readline = require('readline');
const Table = require('cli-table3');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true,
useUnifiedTopology: true });

const Student = mongoose.model('Student', {
    name: String,
    age: Number,
    address: String,
    phoneNumber: String,
    email: String
});
```

```
// Create a function to get user input for the student name to delete
function getStudentNameToDelete() {
    rl.question('Enter the name of the student to delete: ', (studentName) =>
{
        deleteStudentByName(studentName);
    });
}

// Function to delete a student by name and display the deleted student in a
table format
function deleteStudentByName(studentName) {
    Student.findOneAndDelete({ name: studentName })
        .then((deletedStudent) => {
            if (deletedStudent) {
                console.log(`\nStudent with the name "${studentName}"
deleted:`);
                displayTable([deletedStudent]);
            } else {
                console.log(`\nNo student found with the name
"${studentName}".`);
            }
        })
        .catch((err) => {
            console.error(err);
        })
        .finally(() => {
            rl.close(); // Close the readline interface
            mongoose.disconnect(); // Close the database connection
        });
}

// Function to display data in a table format
function displayTable(data) {
    const table = new Table({
        head: ['Name', 'Age', 'Address', 'Phone Number', 'Email'],
        colWidths: [20, 6, 30, 15, 30],
    });

    data.forEach((item) => {
        table.push([item.name, item.age, item.address, item.phoneNumber,
item.email]);
    });

    console.log(table.toString());
}

// Start by getting the name of the student to delete
```

```
getStudentNameToDelete();
```

✓ **Output :**

```
pwsh  811  22ms                                  20.4.0  79  10,12:04
node 4.js
Enter the name of the student to delete: Jack Sparrow

Student with the name "Jack Sparrow" deleted:
```

Name	Age	Address	Phone Number	Email
Jack Sparrow	50	RIO	112366	jack@mail.com

✓ **8_5.js :-**

```
const mongoose = require('mongoose');
const readline = require('readline');
const Table = require('cli-table3');

const rl = readline.createInterface({
    input: process.stdin,
    output: process.stdout
});

mongoose.connect('mongodb://localhost:27017/student', { useNewUrlParser: true,
useUnifiedTopology: true });

const Student = mongoose.model('Student', {
    name: String,
    age: Number,
    address: String,
    phoneNumber: String,
    email: String
});

// Create a function to get user input for the number of students to insert
function getNumberOfStudents() {
    rl.question('Enter the number of students you want to insert: ',
(numStudentsInput) => {
        const numStudents = parseInt(numStudentsInput);
        insertStudents(numStudents);
    });
}

// Create a function to get user input for student details and insert students
function insertStudents(numStudents) {
    const studentsData = [];

    function insertStudent(index) {
```

```
        if (index < numStudents) {
            rl.question(`Enter details for Student ${index + 1}:\nName: `,
(name) => {
                rl.question('Age: ', (age) => {
                    rl.question('Address: ', (address) => {
                        rl.question('Phone Number: ', (phoneNumber) => {
                            rl.question('Email: ', (email) => {
                                studentsData.push({
                                    name: name,
                                    age: parseInt(age),
                                    address: address,
                                    phoneNumber: phoneNumber,
                                    email: email
                                });

                                insertStudent(index + 1); // Insert the next
student
                            });
                        });
                    });
                });
            });
        } else {
            // All students are collected, insert them into the database
            Student.insertMany(studentsData)
                .then((students) => {
                    console.log(`\nInserted ${students.length} students:`);
                    displayTable(students);

                    // Filter students whose age is greater than 16
                    return Student.find({ age: { $gt: 16 } });
                })
                .then((filteredStudents) => {
                    console.log('\nStudents older than 16:');
                    displayTable(filteredStudents);

                    // Delete the student records whose phone numbers contain
the digit '6'
                    return Student.find({ phoneNumber: /6/ });
                })
                .then((studentsToDelete) => {
                    console.log('\nStudents to delete (phone numbers
containing "6"):');
                    displayTable(studentsToDelete);

                    // Delete the student records
                    return Student.deleteMany({ phoneNumber: /6/ });
                })
```

```
                .then(() => {
                    console.log('\nRecords with phone numbers containing "6"
deleted.');
                    rl.close(); // Close the readline interface
                    mongoose.disconnect(); // Close the database connection
                })
                .catch((err) => {
                    console.error(err);
                    rl.close(); // Close the readline interface
                    mongoose.disconnect(); // Close the database connection
                });
        }
    }

    insertStudent(0); // Start inserting students
}

// Function to display data in a table format
function displayTable(data) {
    const table = new Table({
        head: ['Name', 'Age', 'Address', 'Phone Number', 'Email'],
        colWidths: [20, 6, 30, 15, 30],
    });

    data.forEach((item) => {
        table.push([item.name, item.age, item.address, item.phoneNumber,
item.email]);
    });

    console.log(table.toString());
}

// Start by getting the number of students to insert
getNumberOfStudents();
```

✓ Output :

```
pwsh  811  12s 487ms                                  20.4.0  79  10,12:17
node 5.js
Enter the number of students you want to insert: 10
Enter details for Student 1:
Name: Thor
Age: 1669
Address: Asgard
Phone Number: 100
Email: strombreaker@marvel.com
Enter details for Student 2:
Name: Iron Man
Age: 69
Address: New York
Phone Number: 669
Email: stark@marvel.com
Enter details for Student 3:
Name: Captain America
Age: 60
Address: NYC
Phone Number: 116
Email: steve@marvel.com
Enter details for Student 4:
Name: Loki
Age: 16
Address: Asgard
Phone Number: 1126
Email: loki@marvel.com
Enter details for Student 5:
Name: Hulk
Age: 16
Address: NYC
Phone Number: 1136
Email: bruce@marvel.com
Enter details for Student 6:
Name: Thanos
Age: 1000
Address: Titan
Phone Number: 11234
```

Inserted 10 students:

Name	Age	Address	Phone Number	Email
Thor	1669	Asgard	100	strombreaker@marvel.com
Iron Man	69	New York	669	stark@marvel.com
Captain America	60	NYC	116	steve@marvel.com
Loki	16	Asgard	1126	loki@marvel.com
Hulk	16	NYC	1136	bruce@marvel.com
Thanos	1000	Titan	11234	thanos@marvel.com
Wanda	29	New Jersey	1156	wanda@marvel.com
Black Widow	39	San Fransico	1125	natasha@marvel.com
Black Panther	37	Wakanda	007	Tchalla@marvel.com
Doctor Strange	42	New York	11256	Stephen@marvel.com

Students older than 16:

Name	Age	Address	Phone Number	Email
James Bond	20	New York	007	jamesbond@gmail.com
Thor	1669	Asgard	100	strombreaker@marvel.com
Iron Man	69	New York	669	stark@marvel.com
Captain America	60	NYC	116	steve@marvel.com
Thanos	1000	Titan	11234	thanos@marvel.com
Wanda	29	New Jersey	1156	wanda@marvel.com
Black Widow	39	San Fransico	1125	natasha@marvel.com
Black Panther	37	Wakanda	007	Tchalla@marvel.com
Doctor Strange	42	New York	11256	Stephen@marvel.com

Students to delete (phone numbers containing "6"):

Name	Age	Address	Phone Number	Email
Iron Man	69	New York	669	stark@marvel.com
Captain America	60	NYC	116	steve@marvel.com
Loki	16	Asgard	1126	loki@marvel.com
Hulk	16	NYC	1136	bruce@marvel.com
Wanda	29	New Jersey	1156	wanda@marvel.com
Doctor Strange	42	New York	11256	Stephen@marvel.com