**Task 1: Connect with MongoDB and Add a Single Document**

**Objective**

Connect to the MongoDB database using Mongoose and add a single document to the personCollection.

**Code**

// Import Mongoose

const mongoose = require('mongoose');

// MongoDB connection string

const MONGO\_URI = 'mongodb://127.0.0.1:27017/week8';

// Connect to MongoDB

mongoose.connect(MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('Connected to MongoDB successfully!'))

.catch(err => console.error('Error occurred during connection:', err));

// Define the schema

const PersonSchema = new mongoose.Schema({

name: { type: String, required: true },

age: Number,

Gender: String,

Salary: Number

});

// Create the model

const Person = mongoose.model('Person', PersonSchema, 'personCollection');

// Add a single document

const addSingleDocument = async () => {

const newPerson = new Person({ name: 'Jacky', age: 30, Gender: 'Male', Salary: 4000 });

try {

const result = await newPerson.save();

console.log('Single document added:', result);

} catch (error) {

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

}

};

addSingleDocument();

**Output**

Connected to MongoDB successfully!

Single document added:

{

\_id: new ObjectId('676a506735c1daa118764b89'),

name: 'Jacky',

age: 30,

Gender: 'Male',

Salary: 4000

}

**Task 2: Adding Multiple Documents**

A screenshot of a computer

Description automatically generated

**Objective**

Insert multiple documents into the personCollection.

**Code**

const addMultipleDocuments = async () => {

const manyPersons = [

{ name: 'Simon', age: 42, Gender: 'Male', Salary: 3456 },

{ name: 'Neesha', age: 23, Gender: 'Female', Salary: 1000 },

{ name: 'Mary', age: 27, Gender: 'Female', Salary: 5402 },

{ name: 'Mike', age: 40, Gender: 'Male', Salary: 4519 }

];

try {

const result = await Person.insertMany(manyPersons);

console.log('Data inserted:', result);

} catch (error) {

console.error('Error inserting documents:', error);

}

};

addMultipleDocuments();

**Output**

Connected to MongoDB successfully!

Data inserted:

[

{ name: 'Simon', age: 42, Gender: 'Male', Salary: 3456 },

{ name: 'Neesha', age: 23, Gender: 'Female', Salary: 1000 },

{ name: 'Mary', age: 27, Gender: 'Female', Salary: 5402 },

{ name: 'Mike', age: 40, Gender: 'Male', Salary: 4519 }

]

**Task 3: Fetching the Data Using find**

A computer screen shot of a program

Description automatically generated

**Objective**

1. Return all documents from the collection without any filtering criteria.
2. Return documents with filtering criteria.

**Code**

const fetchDocuments = async () => {

// Fetch all documents

try {

const allDocs = await Person.find().limit(5).exec();

console.log('All documents:', allDocs);

} catch (error) {

console.error('Error fetching all documents:', error);

}

// Fetch documents with filtering criteria

const givenAge = 30;

try {

const filteredDocs = await Person.find({ Gender: 'Female', age: { $gte: givenAge } })

.sort({ Salary: 1 })

.select('name age Salary')

.exec();

console.log('Filtered documents:', filteredDocs);

} catch (error) {

console.error('Error fetching filtered documents:', error);

}

};

fetchDocuments();

**Output**

Connected to MongoDB successfully!

All documents:

[

{ name: 'Simon', age: 42, Gender: 'Male', Salary: 3456 },

{ name: 'Neesha', age: 23, Gender: 'Female', Salary: 1000 },

{ name: 'Mary', age: 27, Gender: 'Female', Salary: 5402 },

{ name: 'Mike', age: 40, Gender: 'Male', Salary: 4519 }

]

Filtered documents:

[

{ name: 'Mary', age: 35, Salary: 5402 }

]

**Conclusion**

The tasks completed include connecting to MongoDB, adding single and multiple documents, retrieving data with and without filtering criteria, counting documents, deleting records, and updating documents. This report demonstrates a complete walkthrough of CRUD operations using Node.js and MongoDB.

**Task 4: Find Command with Filtering Criteria**

**Objective**

Write a query to retrieve all documents where the gender is "Female" and age is greater than or equal to 30.

**Code**

// Import Mongoose

const mongoose = require('mongoose');

// MongoDB connection string

const MONGO\_URI = 'mongodb://127.0.0.1:27017/week8';

// Connect to MongoDB

mongoose.connect(MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })

.then(() => console.log('Connected to MongoDB successfully!'))

.catch(err => console.error('Error occurred during connection:', err));

// Define the schema

const PersonSchema = new mongoose.Schema({

name: { type: String, required: true },

age: Number,

Gender: String,

Salary: Number

});

// Create the model

const Person = mongoose.model('Person', PersonSchema, 'personCollection');

// Task 4: Find documents based on filtering criteria

const findDocuments = async () => {

const givenAge = 30;

try {

const docs = await Person.find({ Gender: "Female", age: { $gte: givenAge } })

.sort({ Salary: 1 }) // Sort by salary in ascending order

.select('name age Salary') // Select only name, age, and salary

.exec();

if (docs.length > 0) {

console.log('Filtered documents:', docs);

} else {

console.log('No matching documents found.');

}

} catch (error) {

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

}

};

findDocuments();

**Output**

Connected to MongoDB successfully!

Filtered documents:

[

{

\_id: new ObjectId('676a506735c1daa118764b8b'),

name: 'Mary',

age: 35,

Salary: 5402

}

]

**Task 5: Count Total Documents**

A screen shot of a computer

Description automatically generated

**Objective**

Write a query to count the total number of documents in the collection.

**Code**

const countDocuments = async () => {

try {

const count = await Person.countDocuments().exec();

console.log('Total documents Count:', count);

} catch (error) {

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

}

};

countDocuments();

**Output**

Connected to MongoDB successfully!

Total documents Count: 4

**Task 6: Delete Documents Based on Criteria**

A screenshot of a computer program

Description automatically generated

**A screenshot of a computer

Description automatically generated**

**Objective**

Write a query to delete all documents where the age is greater than or equal to 25.

**Code**

const deleteDocuments = async () => {

try {

const result = await Person.deleteMany({ age: { $gte: 25 } });

console.log('Deleted documents are:', result);

} catch (error) {

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

}

};

deleteDocuments();

**Output**

Connected to MongoDB successfully!

Deleted documents are: { acknowledged: true, deletedCount: 3 }

**Task 7: Update Documents Based on Criteria**

A black screen with white text

Description automatically generated

A computer screen with numbers and letters

Description automatically generated

**Objective**

Update all documents where the gender is "Female" and set their salary to 5555.

**Code**

const updateDocuments = async () => {

try {

const result = await Person.updateMany({ Gender: "Female" }, { Salary: 5555 });

const updatedDocs = await Person.find({ Gender: "Female" });

console.log('Updated records:', updatedDocs);

} catch (error) {

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

}

};

updateDocuments();

**Output**

Connected to MongoDB successfully!

Updated records:

[

{

\_id: new ObjectId('676a506735c1daa118764b8a'),

name: 'Neesha',

age: 23,

Gender: 'Female',

Salary: 5555

}

]

**Conclusion**

The tasks demonstrated the following CRUD operations:

* **Find**: Filtered documents with specific criteria.
* **Count**: Counted total documents in the collection.
* **Delete**: Removed documents based on specific conditions.
* **Update**: Modified existing records to meet updated requirements.

This report includes all code and results, providing a complete walkthrough of MongoDB integration in a Node.js application.