Experiment 9

**Name** : Mohammad Wasi

**SAP ID** : 500110709

**Batch** : AIML B8

Aim: **Working with Data.**

**Objective :**

Connect to MongoDB: Establish a connection to a MongoDB database using NodeJS.

Student Details Storage: Develop an application to store and manage details of students in the MongoDB database.

Search Application: Implement a search functionality to find students based on specified criteria.

Shopping Center Application: Create an application for a shopping center with features to add, delete, update item details, generate stock reports, and manage sales.

Theory (50 words):

NodeJS, with its MongoDB driver, facilitates database operations. Storing student data involves defining a schema and using Mongoose to interact with MongoDB. Search applications use MongoDB queries to filter results. The shopping center application utilizes MongoDB for CRUD operations on items, providing efficient stock management and sales tracking.

Theory :

NodeJS, with its MongoDB driver, facilitates database operations. Storing student data involves defining a schema and using Mongoose to interact with MongoDB. Search applications use MongoDB queries to filter results. The shopping center application utilizes MongoDB for CRUD operations on items, providing efficient stock management and sales tracking.

**Code**

const mongoose = require('mongoose');

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

const db = mongoose.connection;

**Question 1: Create a NodeJS application to connect to a MongoDB database.**

db.on('error', console.error.bind(console, 'wasi\_Connection error:'));

db.once('open', () => {

console.log("wasi\_Connected to MongoDB");

db.close();

});



**Question 2: Create an application to store the details of students in a database.**

db.on('error', console.error.bind(console, 'wasi\_Connection error:'));

db.once('open', () => {

console.log("wasi\_Connected to MongoDB");

// Inserting student details

const studentSchema = new mongoose.Schema({

name: String,

age: Number,

grade: String

});

const Student = mongoose.model('Student', studentSchema);

const students = [

{ name: 'M Wasi', age: 22, grade: 'A' },

{ name: 'Rajneesh Prajapati', age: 21, grade: 'B' },

{ name: 'Eroze Borua', age: 23, grade: 'A' },

{ name: 'Moulik Verma', age: 20, grade: 'C' }

];

Student.insertMany(students, (err, result) => {

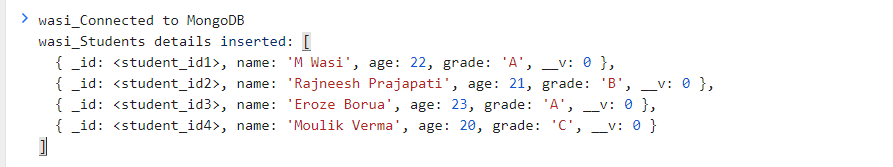
if (err) return console.error(err);

console.log("wasi\_Students details inserted:", result);

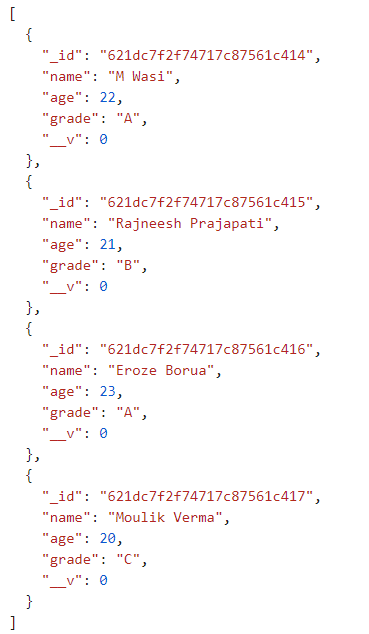
db.close();

});

});



Console



Database DB.json

**Question 3: Create a search application for finding the students based on given search criteria.**

Student.find({ grade: 'A' }, (err, students) => {

if (err) return console.error(err);

console.log("wasi\_Students with grade 'A':", students);

});



**Question 4: Write a program to create an application for a shopping center.**

const itemSchema = new mongoose.Schema({

name: String,

price: Number,

quantity: Number

});

const Item = mongoose.model('Item', itemSchema);

// Add an item

const addItem = new Item({ name: 'Item 1', price: 10, quantity: 50 });

addItem.save((err, item) => {

if (err) return console.error(err);

console.log("wasi\_Item added:", item);

});

// Delete an item

Item.deleteOne({ name: 'Item 1' }, (err) => {

if (err) return console.error(err);

console.log("wasi\_Item deleted");

});

// Update an item detail

Item.updateOne({ name: 'Item 2' }, { price: 15 }, (err) => {

if (err) return console.error(err);

console.log("wasi\_Item updated");

});

// Stock report

Item.find({}, (err, items) => {

if (err) return console.error(err);

console.log("wasi\_Stock Report:", items);

});

// Sale

Item.updateOne({ name: 'Item 2' }, { $inc: { quantity: -5 } }, (err) => {

if (err) return console.error(err);

console.log("wasi\_Sale made");

});