**Experiment : 3 Setting up a MongoDB Database (Connecting MongoDB to your application)**

**Step 1: Install MongoDB**

1. Download MongoDB:
   * Visit the official MongoDB website and download the appropriate version for your operating system.
2. Install MongoDB:
   * Follow the installation instructions provided for your operating system.
   * For detailed installation steps, refer to the MongoDB installation documentation.

**Step 2: Start MongoDB Server**

1. Run MongoDB Server:
   * After installation, start the MongoDB server. The process might vary depending on your operating system.
   * On Unix-based systems, you can start MongoDB using the following command: **mongod**
   * On Windows, you might need to start it as a service or use the mongod command in a command prompt.

**Step 3: Connect to MongoDB**

1. MongoDB Shell:
   * Open a new terminal or command prompt and run the **mongo** command to start the MongoDB shell.
2. Create a Database:
   * In the MongoDB shell, you can create a new database using the **use** command.

**use mydatabase**

Replace "mydatabase" with the desired name for your database.

**Step 4: Connect from your Application**

1. Install MongoDB Driver:
   * Install the MongoDB driver for your programming language. MongoDB provides drivers for various languages, including Node.js, Python, Java, etc.
   * For example, if you're using Node.js, you can install the official MongoDB driver using npm:

**npm install mongodb**

1. Connect in Your Application:
   * Use the appropriate MongoDB connection string in your application to connect to the MongoDB server. The connection string typically includes information such as host, port, username, password, and database name.

Example (Node.js):

const { MongoClient } = require('mongodb');

const uri = 'mongodb://localhost:27017/mydatabase';

const client = new MongoClient(uri, { useNewUrlParser: true, useUnifiedTopology: true });

async function connectToMongoDB() {

try {

await client.connect();

console.log('Connected to MongoDB');

} catch (error) {

console.error('Error connecting to MongoDB:', error);

}

}

connectToMongoDB();

Replace 'mongodb://localhost:27017/mydatabase' with your actual MongoDB connection string.

**Step 5: Perform Database Operations**

Now that you're connected, you can perform various database operations in your application, such as inserting, updating, querying, and deleting documents.

Example (Node.js):

// Assuming 'client' is the MongoDB client obtained in the previous step

**INSERT ONE**

async function inserto(client, newdoc){

    const result = await client.db("krishna").collection("awt").insertOne(newdoc);

    console.log(`New document created with the following id: ${result.insertedId}`);

}

inserto(client,

    {

        name: "krishna",

        division: "5",

        subject: 1,

        classes: 1

    }

);

inserto(client,

    {

        name: "jane",

        division: "6",

        subject: 2,

        classes: 4

    }

);

**INSERT MANY**

async function insertm(client, newdocs){

    const result = await client.db("krishna").collection("awt").insertMany(newdocs);

    console.log(`${result.insertedCount} new documents created with the following id(s):`);

    console.log(result.insertedIds);

}

insertm(client, [

    {

        name: "Infinite Views",

        property\_type: "House",

        bedrooms: 5,

        bathrooms: 4.5,

        beds: 5

    },

    {

        name: "Private room in London",

        property\_type: "Apartment",

        bedrooms: 1,

        bathroom: 1

    },

    {

        name: "Beautiful Beach House",

        bedrooms: 4,

        bathrooms: 2.5,

        beds: 7,

        last\_review: new Date()

    }

]);

**FIND ONE BY A GIVEN QUERY: Here name**

async function findbn(client, nameOfdoc){

    const result = await client.db("krishna").collection("awt").findOne({name: nameOfdoc});

    if (result) {

        console.log(`Found a document in the collection with the name '${nameOfdoc}':`);

        console.log(result);

    } else {

        console.log(`No documents found with the name '${nameOfdoc}'`);

    }

}

 findbn(client, "Infinite Views");

**Similarly any filters can be added.**

**UPDATE ONE**

async function updatedocbn(client, nameOfdoc, updateddoc) {

    const result = await client.db("krishna").collection("awt").updateOne({ name: nameOfdoc }, { $set: updateddoc });

    console.log(`${result.matchedCount} document(s) matched the query.`);

    console.log(`${result.modifiedCount} document(s) was/were updated.`);

}

updatedocbn(client, "Infinite Views", { bedrooms: 6, beds: 8 });

**UPDATE MANY**

async function updatedmbn(client, nameOfdoc, updateddoc) {

    const result = await client.db("krishna").collection("awt").updateMany({ name: nameOfdoc }, { $set: updateddoc });

    console.log(`${result.matchedCount} document(s) matched the query.`);

    console.log(`${result.modifiedCount} document(s) was/were updated.`);

}

updatedmbn(client, "Infinite Views", { bedrooms: 2, beds: 1 });

**DELETE ONE**

async function deleteobn(client, nameOfdoc) {

    const result = await client.db("krishna").collection("awt").deleteOne({ name: nameOfdoc });

    console.log(`${result.deletedCount} document(s) deleted by the query.`);

}

deleteobn(client, "jane");

**DELETE MANY**

async function deletembn(client, nameOfdoc) {

    const result = await client.db("krishna").collection("awt").deleteMany({ name: nameOfdoc });

    console.log(`${result.deletedCount} document(s) deleted by the query.`);

}

deletembn(client, "Infinite Views");

**Through plugin**

**Ctrl+shift+x**

**Mongodb for vscode install**

**You see an new bar of mongodb**

**Do connection**

**A new playground created**

**If not go to terminal and ctrl+shift+p search mongodb see comands**

db.getCollection('sales').insertMany([

  { 'item': 'abc', 'price': 10, 'quantity': 2, 'date': new Date('2014-03-01T08:00:00Z') },

  { 'item': 'jkl', 'price': 20, 'quantity': 1, 'date': new Date('2014-03-01T09:00:00Z') },

  { 'item': 'xyz', 'price': 5, 'quantity': 10, 'date': new Date('2014-03-15T09:00:00Z') },

  { 'item': 'xyz', 'price': 5, 'quantity': 20, 'date': new Date('2014-04-04T11:21:39.736Z') },

  { 'item': 'abc', 'price': 10, 'quantity': 10, 'date': new Date('2014-04-04T21:23:13.331Z') },

  { 'item': 'def', 'price': 7.5, 'quantity': 5, 'date': new Date('2015-06-04T05:08:13Z') },

  { 'item': 'def', 'price': 7.5, 'quantity': 10, 'date': new Date('2015-09-10T08:43:00Z') },

  { 'item': 'abc', 'price': 10, 'quantity': 5, 'date': new Date('2016-02-06T20:20:13Z') },

]);

// Run a find command to view items sold on April 4th, 2014.

const salesOnApril4th = db.getCollection('sales').find({

  date: { $gte: new Date('2014-04-04'), $lt: new Date('2014-04-05') }

}).count();

// Print a message to the output window.

console.log(`${salesOnApril4th} sales occurred in 2014.`);

const data =  db.getCollection('awt').find({})

console.log(data);