const prompt = require('prompt-sync')();

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

class Leg {

constructor(source, destination, cost) {

this.source = source;

this.destination = destination;

this.cost = cost;

}

}

class Route {

constructor() {

this.legs = [];

this.collectionName = '2'; // Name of the MongoDB collection

}

async connect() {

this.client = await MongoClient.connect("mongodb+srv://Niran\_jana\_2604:12345@cluster0.k9vdgjh.mongodb.net/?retryWrites=true&w=majority", {

useUnifiedTopology: true,

});

this.db = this.client.db('day'); // Replace with your database name

this.collection = this.db.collection(this.collectionName);

}

async disconnect() {

await this.client.close();

}

async addLeg(source, destination, cost) {

const leg = new Leg(source, destination, cost);

this.legs.push(leg);

await this.collection.insertOne(leg);

}

async getTotalCost() {

return this.legs.reduce((totalCost, leg) => totalCost + leg.cost, 0);

}

async updateLeg(legId, updatedLeg) {

const legObjectId = new ObjectId(legId);

await this.collection.updateOne(

{ \_id: legObjectId },

{ $set: updatedLeg }

);

}

async deleteLeg(legId) {

const legObjectId = new ObjectId(legId);

await this.collection.deleteOne({ \_id: legObjectId });

}

async getLegs() {

return await this.collection.find().toArray();

}

}

async function main() {

const route = new Route();

await route.connect();

const numLegs = parseInt(prompt('Enter the number of legs in the route:'));

for (let i = 1; i <= numLegs; i++) {

const source = prompt(`Enter the source city for leg ${i}:`);

const destination = prompt(`Enter the destination city for leg ${i}:`);

const cost = parseFloat(prompt(`Enter the cost for leg ${i}:`));

await route.addLeg(source, destination, cost);

}

const totalCost = await route.getTotalCost();

console.log('Total cost of the trip:', totalCost);

let isRunning = true;

while (isRunning) {

console.log('\nChoose a CRUD operation:');

console.log('1. Add leg');

console.log('2. Update leg');

console.log('3. Delete leg');

console.log('4. Get all legs');

console.log('5. Exit');

const option = prompt('Enter your choice: ');

switch (option) {

case '1': {

const source = prompt('Enter the source city: ');

const destination = prompt('Enter the destination city: ');

const cost = parseFloat(prompt('Enter the cost: '));

await route.addLeg(source, destination, cost);

console.log('Leg added successfully.');

break;

}

case '2': {

const legId = prompt('Enter the ID of the leg to update: ');

const source = prompt('Enter the updated source city: ');

const destination = prompt('Enter the updated destination city: ');

const cost = parseFloat(prompt('Enter the updated cost: '));

const updatedLeg = new Leg(source, destination, cost);

await route.updateLeg(legId, updatedLeg);

console.log('Leg updated successfully.');

break;

}

case '3': {

const legId = prompt('Enter the ID of the leg to delete: ');

await route.deleteLeg(legId);

console.log('Leg deleted successfully.');

break;

}

case '4': {

const legs = await route.getLegs();

console.log('All legs:');

console.log(legs);

break;

}

case '5': {

isRunning = false;

break;

}

default: {

console.log('Invalid option. Please try again.');

break;

}

}

}

await route.disconnect();

}

main();