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

const mongoose = require('mongoose');

// Connect to MongoDB

mongoose.connect('mongodb+srv://Niran\_jana\_2604:12345@cluster0.k9vdgjh.mongodb.net/day?retryWrites=true&w=majority', {

useNewUrlParser: true,

useUnifiedTopology: true,

});

// Define Leg schema

const legSchema = new mongoose.Schema({

source: String,

destination: String,

cost: Number,

});

// Define Leg model

const Leg = mongoose.model('Leg', legSchema);

class Route {

constructor() {

this.legs = [];

}

async addLeg(source, destination, cost) {

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

await leg.save();

this.legs.push(leg);

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

}

getTotalCost() {

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

}

async updateLeg(legIndex, source, destination, cost) {

const leg = this.legs[legIndex];

leg.source = source;

leg.destination = destination;

leg.cost = cost;

await leg.save();

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

}

async deleteLeg(legIndex) {

const leg = this.legs[legIndex];

await Leg.deleteOne({ \_id: leg.\_id });

this.legs.splice(legIndex, 1);

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

}

async getAllLegs() {

const legs = await Leg.find({});

console.log('All legs:');

console.log(legs);

}

}

async function main() {

const route = new Route();

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 = 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);

break;

}

case '2': {

const legIndex = parseInt(prompt('Enter the index 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: '));

await route.updateLeg(legIndex, source, destination, cost);

break;

}

case '3': {

const legIndex = parseInt(prompt('Enter the index of the leg to delete: '));

await route.deleteLeg(legIndex);

break;

}

case '4': {

await route.getAllLegs();

break;

}

case '5': {

isRunning = false;

break;

}

default: {

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

break;

}

}

}

// Disconnect from MongoDB

await mongoose.disconnect();

}

main();