Exp 10

**Title**

CLI Employee Management System Using Node.js and Arrays

**Objective**

Learn how to build an interactive command-line interface (CLI) application using Node.js that manages data in memory with arrays. This task strengthens your understanding of basic Node.js, user input handling, and array operations.

**Task Description**

Create a simple Node.js CLI application that manages a list of employees stored in an array. The app should allow the user to perform actions like adding a new employee (with name and ID), listing all employees, and removing an employee by ID. Use built-in Node.js modules like readline to capture user input interactively in the terminal. All data should be stored and updated directly in an array during the session, without using any external databases or files.

Code

const readline = require("readline");

const rl = readline.createInterface({

  input: process.stdin,

  output: process.stdout,

});

let employees = [];

function showMenu() {

  console.log("\nEmployee Management System");

  console.log("1. Add Employee");

  console.log("2. List Employees");

  console.log("3. Search Employee");

  console.log("4. Update Employee");

  console.log("5. Delete Employee");

  console.log("6. Exit");

  rl.question("Choose an option: ", handleMenu);

}

function handleMenu(choice) {

  switch (choice.trim()) {

    case "1":

      addEmployee();

      break;

    case "2":

      listEmployees();

      break;

    case "3":

      searchEmployee();

      break;

    case "4":

      updateEmployee();

      break;

    case "5":

      deleteEmployee();

      break;

    case "6":

      rl.close();

      break;

    default:

      console.log("Invalid choice");

      showMenu();

  }

}

// Promise for async input

function ask(question) {

  return new Promise((resolve) => rl.question(question, resolve));

}

async function addEmployee() {

  const name = await ask("Enter name: ");

  const position = await ask("Enter position: ");

  const salary = parseInt(await ask("Enter salary: "));

  let id = employees.length + 1;

  employees.push({ id, name, position, salary });

  console.log("Employee added successfully!");

  showMenu();

}

function listEmployees() {

  if (employees.length === 0) {

    console.log("No employees found.");

  } else {

    console.log("\nEmployees:");

    employees.forEach((emp) => {

      console.log(`${emp.id}. ${emp.name} - ${emp.position} - $${emp.salary}`);

    });

  }

  showMenu();

}

async function searchEmployee() {

  const query = await ask("Enter employee ID or name: ");

  const found = employees.find(

    (emp) =>

      emp.id === parseInt(query) ||

      emp.name.toLowerCase() === query.toLowerCase()

  );

  if (found) {

    console.log(`Found: ${found.id}. ${found.name} - ${found.position} -

$${found.salary}`);

  } else {

    console.log("Employee not found.");

  }

  showMenu();

}

async function updateEmployee() {

  const id = await ask("Enter employee ID to update: ");

  const emp = employees.find((e) => e.id === parseInt(id));

  if (!emp) {

    console.log("Employee not found.");

    return showMenu();

  }

  const name = await ask(`Enter new name (${emp.name}): `);

  const position = await ask(`Enter new position (${emp.position}): `);

  const salary = await ask(`Enter new salary (${emp.salary}): `);

  if (name) emp.name = name.trim();

  if (position) emp.position = position.trim();

  if (salary) emp.salary = parseFloat(salary);

  console.log("Employee updated successfully!");

  showMenu();

}

async function deleteEmployee() {

  const id = await ask("Enter employee ID to delete: ");

  const index = employees.findIndex((e) => e.id === parseInt(id));

  if (index !== -1) {

    employees.splice(index, 1);

    console.log("Employee deleted successfully!");

  } else {

    console.log("Employee not found.");

  }

  showMenu();

}

showMenu();

output

