// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract EmployeeData

    // Structure to hold employee details

    struct Employee {

        uint id;

        string name;

        string department;

        uint salary; }

    // Array to store employee data

    Employee[] public employees;

    // Mapping to keep track of employee existence

    mapping(uint => bool) public employeeExists;

    // Function to add a new employee

    function addEmployee(uint \_id, string memory \_name, string memory \_department, uint \_salary) public {

        require(!employeeExists[\_id], "Employee ID already exists.");

        employees.push(Employee(\_id, \_name, \_department, \_salary));

        employeeExists[\_id] = true; }

    // Function to get the total number of employees

    function getEmployeeCount() public view returns (uint) {

        return employees.length; }

    // Function to retrieve employee details by id

    function getEmployee(uint \_id) public view returns (string memory name, string memory department, uint salary) {

        for (uint i = 0; i < employees.length; i++) {

            if (employees[i].id == \_id) {

                return (employees[i].name, employees[i].department, employees[i].salary) }}

        revert("Employee not found")  }  // Fallback function to handle calls to non-existent functions or Ether with data

    fallback() external payable {

        // Ether sent with data will be stored her }

    // Receive function to handle plain Ether transfers

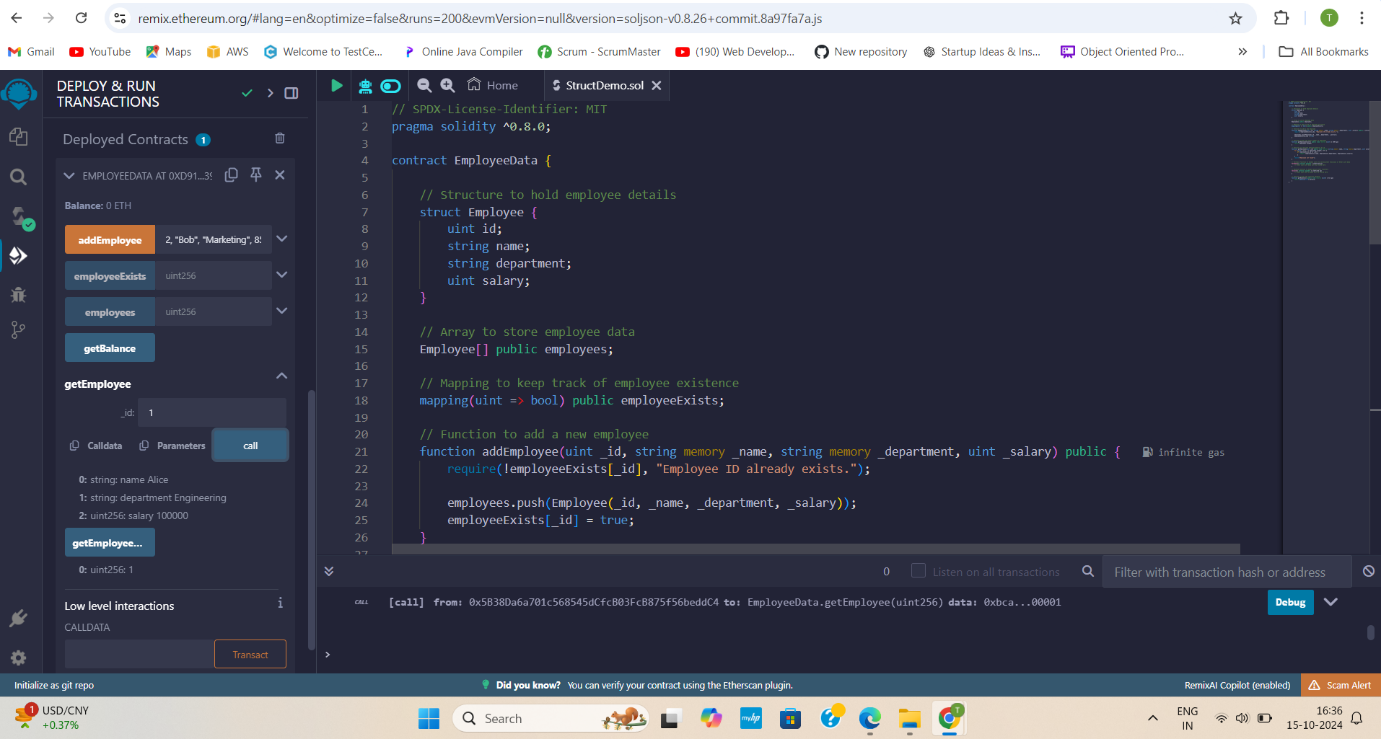
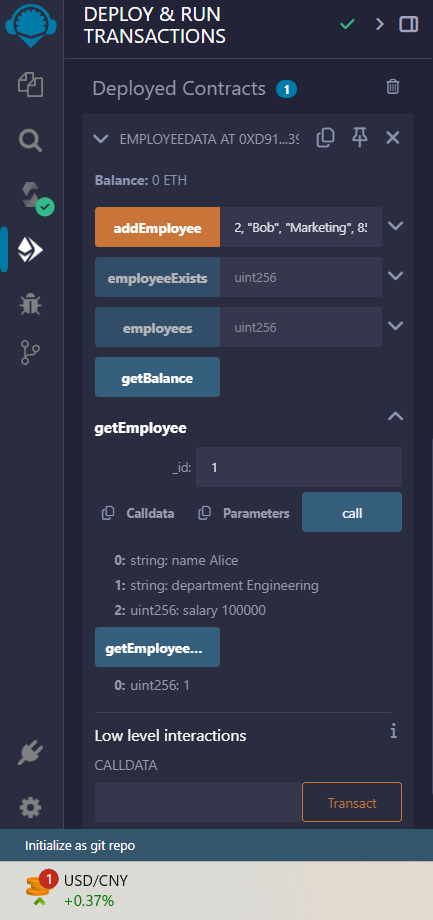
    receive() external payable {

        // Ether sent with no data will be stored here }

    // Function to get the contract balance

    function getBalance() public view returns (uint) {

        return address(this).balance; }}

****