package com.example.junit\_testing.service;

import com.example.junit\_testing.dto.EmployeeDto;

import com.example.junit\_testing.entity.Employee;

import com.example.junit\_testing.repository.EmployeeRepository;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.Test;

import org.junit.jupiter.api.extension.ExtendWith;

import org.mockito.InjectMocks;

import org.mockito.Mock;

import org.mockito.Mockito;

import org.mockito.junit.jupiter.MockitoExtension;

import java.util.Arrays;

import java.util.List;

import java.util.Optional;

import static org.junit.jupiter.api.Assertions.\*;

import static org.mockito.Mockito.\*;

@ExtendWith(MockitoExtension.class)

class EmployeeServiceTest {

@Mock

private EmployeeRepository employeeRepository;

@InjectMocks

private EmployeeService employeeService;

private Employee employee;

private EmployeeDto employeeDto;

@BeforeEach

void setUp() {

employee = Employee.builder()

.id(1L)

.name("John Doe")

.age(30)

.salary(50000)

.build();

employeeDto = new EmployeeDto("John Doe", 30, 50000);

}

@Test

void testGetAllEmployee() {

when(employeeRepository.findAll()).thenReturn(List.of(employee));

List<Employee> employees = employeeService.getAllEmployee();

assertEquals(1, employees.size());

assertEquals("John Doe", employees.get(0).getName());

verify(employeeRepository, times(1)).findAll();

}

@Test

void testAddEmployeeDetails() {

when(employeeRepository.save(Mockito.any(Employee.class))).thenReturn(employee);

Employee savedEmployee = employeeService.addEmployeeDetails(employeeDto);

assertNotNull(savedEmployee);

assertEquals("John Doe", savedEmployee.getName());

verify(employeeRepository, times(1)).save(any(Employee.class));

}

@Test

void testGetEmployeeByName\_Success() {

when(employeeRepository.findByName("John Doe")).thenReturn(List.of(employee));

List<Employee> employees = employeeService.getEmployeeByName("John Doe");

assertFalse(employees.isEmpty());

assertEquals("John Doe", employees.get(0).getName());

verify(employeeRepository, times(1)).findByName("John Doe");

}

@Test

void testGetEmployeeByName\_NotFound() {

when(employeeRepository.findByName("Jane Doe")).thenReturn(List.of());

Exception exception = assertThrows(NullPointerException.class, () ->

employeeService.getEmployeeByName("Jane Doe"));

assertEquals("there is no records", exception.getMessage());

verify(employeeRepository, times(1)).findByName("Jane Doe");

}

}