package com.vikram.service;

import java.util.ArrayList;

import java.util.List;

import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import com.vikram.dto.UserDto;

import com.vikram.entity.Customer;

import com.vikram.entity.Engineer;

import com.vikram.entity.Manager;

import com.vikram.entity.User;

import com.vikram.entity.Zipcode;

import com.vikram.repository.CustomerRepository;

import com.vikram.repository.EngineerRepository;

import com.vikram.repository.ManagerRepository;

import com.vikram.repository.UserRepository;

import com.vikram.repository.ZipcodeRepository;

@Service

public class UserServiceImpl implements UserService {

@Autowired

private CustomerRepository customerRepository;

@Autowired

private EngineerRepository engineerRepository;

@Autowired

private ManagerRepository managerRepository;

@Autowired

private ZipcodeRepository zipcodeRepository;

@Autowired

private UserRepository userRepository;

// Password encoder will be added after adding the security dependency

// private BCryptPasswordEncoder passwordEncoder;

@Override

public List<String> getZipcodes(){

List<String> zipcodes = new ArrayList<String>();

zipcodeRepository.findAll().stream().forEach(zipcode->zipcodes.add(zipcode.getZipcode()));

return zipcodes;

}

@Override

public String addCustomer(Customer customer) {

return customerRepository.save(customer).getId().toString();

}

@Override

public String addEngineer(Engineer engineer) {

return engineerRepository.save(engineer).getId().toString();

}

@Override

public String addManager(Manager manager) {

return managerRepository.save(manager).getId().toString();

}

@Override

public List<Customer> getAllCustomers() {

return customerRepository.findAll();

}

@Override

public List<Engineer> getAllEngineers() {

return engineerRepository.findAll();

}

@Override

public List<Manager> getAllManagers() {

return managerRepository.findAll();

}

@Override

public Customer getCustomerById(long id) {

Optional<Customer> optional= customerRepository.findById(id);

if(optional.isPresent())

return optional.get();

return new Customer();

}

@Override

public Engineer getEngineerById(long id) {

Optional<Engineer> optional = engineerRepository.findById(id);

return optional.get();

}

@Override

public Manager getManagerById(long id) {

return managerRepository.findById(id).get();

}

@Override

public String deleteCustomer(Long id) {

customerRepository.delete(getCustomerById(id));

return id.toString();

}

@Override

public User loginCheck(UserDto userDto) {

User user;

List<User> users = userRepository.findByEmail(userDto.getEmail());

if(!users.isEmpty()) {

user = users.get(0);

if(user.getPassword().equals(userDto.getPassword())) {

return user;

}

}

return null;

}

@Override

public Customer findCustomerByUserId(Long id) {

User user = userRepository.findById(id).get();

return customerRepository.findByUser(user).get(0);

}

@Override

public Engineer findEngineerByUserId(Long id) {

User user = userRepository.findById(id).get();

return engineerRepository.findByUser(user).get(0);

}

@Override

public Manager findManagerByUserId(Long id) {

User user = userRepository.findById(id).get();

return managerRepository.findByUser(user).get(0);

}

@Override

public List<Engineer> getEngineersByZipcode(String zipcode) {

return engineerRepository.findByZipcode(zipcode);

}

}