package com.icinbank.service.impl;

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

import org.springframework.stereotype.Service;

import com.icinbank.dao.AccountRepository;

import com.icinbank.dao.SaccountRepository;

import com.icinbank.dao.UserRepository;

import com.icinbank.model.Account;

import com.icinbank.model.Saccount;

import com.icinbank.model.User;

import com.icinbank.response.DepositResponse;

import com.icinbank.response.TransferResponse;

import com.icinbank.response.WithdrawResponse;

import com.icinbank.service.AccountService;

import com.icinbank.service.TransferHistoryService;

import com.icinbank.service.UserHistoryService;

@Service

public class AccountServiceImpl implements AccountService{

@Autowired

private AccountRepository dao;

@Autowired

private UserHistoryService service;

@Autowired

private TransferHistoryService tservice;

@Autowired

private UserRepository udao;

@Autowired

private SaccountRepository sdao;

private final String bankCode = "3914";

private final String countryCode = "91";

private final String branchCode = "820";

private final String accountcode="1";

public long generate\_saving(int userId) {

String accNo = bankCode+countryCode+branchCode+accountcode+String.valueOf(userId);

return Long.parseLong(accNo);

}

public static boolean isprimary(long account) {

String s = Long.toString(account).substring(0, 10);

String check="3914918201";

if(s.equals(check)) {

return true;

}

else

{

return false;

}

}

@Override

public Account newAccount(String username,int userId) {

Account account=new Account();

account.setUsername(username);

account.setAccno(generate\_saving(userId));

account.setUser(udao.findByUsername(username));

return dao.save(account);

}

@Override

public Account getAccount(String username) {

// TODO Auto-generated method stub

return dao.findByUsername(username);

}

@Override

public DepositResponse deposit(long acc, int amount) {

DepositResponse response=new DepositResponse();

boolean flag=true;

try {

Account account=dao.findByAccno(acc);

account.setBalance(account.getBalance()+amount);

service.addAction(acc, amount, account.getBalance(), "credit");

dao.save(account);

response.setResponseMessage("Rs."+amount+" successfully deposited into your account balance is now Rs."+account.getBalance());

response.setDepositStatus(flag);

}

catch (Exception e) {

flag=false;

response.setResponseMessage("Account number is incorrect");

response.setDepositStatus(flag);

}

response.setAccount(acc);

return response;

}

@Override

public WithdrawResponse withdraw(long acc, int amount) {

WithdrawResponse response=new WithdrawResponse();

boolean flag=true;

try {

Account account=dao.findByAccno(acc);

User user=udao.findByUsername(account.getUsername());

if(user.getFeatureStatus()==2 || user.getFeatureStatus()==3)

{

if(account.getBalance()>=amount)

{

account.setBalance(account.getBalance()-amount);

service.addAction(acc, amount, account.getBalance(), "debit");

dao.save(account);

response.setResponseMessage("Rs."+amount+" successfully withdrawn your account balance is now Rs."+account.getBalance());

response.setWithdrawStatus(flag);

}

else

{

flag=false;

response.setResponseMessage("Insufficient funds to complete the transaction");

response.setWithdrawStatus(flag);

}

}

else {

flag=false;

response.setResponseMessage("This function is not available for your account");

response.setWithdrawStatus(flag);

}

}

catch (Exception e) {

flag=false;

response.setResponseMessage("Account number is incorrect");

response.setWithdrawStatus(flag);

}

response.setAccount(acc);

return response;

}

@Override

public TransferResponse transfer(long saccount, long raccount, int amount) {

TransferResponse response=new TransferResponse();

boolean flag=true;

try {

Account senderAccount=dao.findByAccno(saccount);

//String check = Long.toString(raccount);

if(isprimary(raccount))

{

Account receiverAccount=dao.findByAccno(raccount);

if(senderAccount.getAccno()!=receiverAccount.getAccno())

{

if(senderAccount.getBalance()>amount) {

User user=udao.findByUsername(senderAccount.getUsername());

if(user.getFeatureStatus()==3)

{

senderAccount.setBalance(senderAccount.getBalance()-amount);

receiverAccount.setBalance(receiverAccount.getBalance()+amount);

tservice.addAction(saccount, raccount, amount);

dao.save(senderAccount);

dao.save(receiverAccount);

response.setResponseMessage("Rs."+amount+" successfully transferred to account "+receiverAccount.getAccno());

response.setTransferStatus(flag);

}

else {

flag=false;

response.setResponseMessage("This feature is not available for your account");

response.setTransferStatus(flag);

}

}

else {

flag=false;

response.setResponseMessage("Insufficient funds to complete the transfer");

response.setTransferStatus(flag);

}

}

else {

flag=false;

response.setResponseMessage("sender and recieiver accounts are same");

response.setTransferStatus(flag);

}

}

else {

Saccount receiverAccount=sdao.findByAccno(raccount);

if(senderAccount.getAccno()!=receiverAccount.getAccno())

{

if(senderAccount.getBalance()>amount) {

User user=udao.findByUsername(senderAccount.getUsername());

if(user.getFeatureStatus()==3)

{

senderAccount.setBalance(senderAccount.getBalance()-amount);

receiverAccount.setBalance(receiverAccount.getBalance()+amount);

tservice.addAction(saccount, raccount, amount);

dao.save(senderAccount);

sdao.save(receiverAccount);

response.setResponseMessage("Rs."+amount+" successfully transferred to account "+receiverAccount.getAccno());

response.setTransferStatus(flag);

}

else {

flag=false;

response.setResponseMessage("This function isnt available for the account");

response.setTransferStatus(flag);

}

}

else {

flag=false;

response.setResponseMessage("Insufficient funds to complete the transfer");

response.setTransferStatus(flag);

}

}

else {

flag=false;

response.setResponseMessage("sender and recieiver accounts are same");

response.setTransferStatus(flag);

}

}

}

catch (Exception e) {

flag=false;

response.setResponseMessage("Account number is incorrect");

response.setTransferStatus(flag);

}

response.setSaccount(saccount);

return response;

}

@Override

public Account getAccountDetails(long account) {

// TODO Auto-generated method stub

return dao.findByAccno(account);

}

@Override

public Account updateAccount(Account account) {

// TODO Auto-generated method stub

return dao.save(account);

}

}