package com.icinbank.controller;

import java.util.Collections;

import java.util.List;

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

import org.springframework.web.bind.annotation.CrossOrigin;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.icinbank.dao.AccountRepository;

import com.icinbank.dao.SaccountRepository;

import com.icinbank.details.TransactionDetails;

import com.icinbank.details.TransferDetails;

import com.icinbank.model.Account;

import com.icinbank.model.Saccount;

import com.icinbank.model.Transfer;

import com.icinbank.model.UserHistory;

import com.icinbank.response.DepositResponse;

import com.icinbank.response.TransferResponse;

import com.icinbank.response.WithdrawResponse;

import com.icinbank.service.AccountService;

import com.icinbank.service.SaccountService;

import com.icinbank.service.TransferHistoryService;

import com.icinbank.service.UserHistoryService;

@RestController

//@CrossOrigin(origins = "http://localhost:4200")

@CrossOrigin(origins = "\*", allowedHeaders = "\*")

public class AccountController {

@Autowired

private AccountService service;

@Autowired

private SaccountService sservice;

@Autowired

private UserHistoryService hservice;

@Autowired

private TransferHistoryService tservice;

@Autowired

private AccountRepository adao;

@Autowired

private SaccountRepository sdao;

private final String ifsc="ICIN7465";

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;

}

}

@GetMapping("/account/details/{account}")

public Account getAccountDetails(@PathVariable("account") int account ) {

return service.getAccountDetails(account);

}

@PutMapping("/account/profile")

public Account updateProfile(@RequestBody Account account) {

return service.updateAccount(account);

}

@GetMapping("/account/getprimary/{username}")

public Account getPrimarydetails(@PathVariable("username") String username) {

return service.getAccount(username);

}

@GetMapping("/account/getsaving/{username}")

public Saccount getSavingdetails(@PathVariable("username") String username) {

return sservice.getAccount(username);

}

@PostMapping("/account/deposit")

public DepositResponse deposit(@RequestBody TransactionDetails details) {

//adao.findByUsername(adao.findByAccno(details.getAccount()).getUsername());

if(isprimary(details.getAccount())) {

return service.deposit(details.getAccount(), details.getAmount());

}

else {

return sservice.deposit(details.getAccount(), details.getAmount());

}

}

@PostMapping("/account/withdraw")

public WithdrawResponse withdraw(@RequestBody TransactionDetails details) {

if(isprimary(details.getAccount())) {

return service.withdraw(details.getAccount(), details.getAmount());

}

else {

return sservice.withdraw(details.getAccount(), details.getAmount());

}

}

@PostMapping("/account/transfer")

public TransferResponse transfer(@RequestBody TransferDetails details) {

try {

if(details.getIfsc().equals(ifsc))

{

Account p=adao.findByUsername(details.getUsername());

Saccount s=sdao.findByUsername(details.getUsername());

if(p.getAccno()==details.getSaccount() || s.getAccno()==details.getSaccount()) {

//String len = Integer.toString(details.getSaccount());

if(isprimary(details.getSaccount())) {

return service.transfer(details.getSaccount(), details.getRaccount(), details.getAmount());

}

else

{

return sservice.transfer(details.getSaccount(), details.getRaccount(), details.getAmount());

}

}

else {

TransferResponse response=new TransferResponse();

response.setSaccount(details.getSaccount());

response.setResponseMessage("Dear user You can only transfer funds from the accounts registed with you");

response.setTransferStatus(false);

return response;

}

}

else {

TransferResponse response=new TransferResponse();

response.setSaccount(details.getSaccount());

response.setResponseMessage("IFSC code is incorrect");

response.setTransferStatus(false);

return response;

}

} catch (Exception e) {

TransferResponse response=new TransferResponse();

response.setSaccount(details.getSaccount());

response.setResponseMessage("Please provide an IFSC code");

response.setTransferStatus(false);

return response;

}

}

@GetMapping("/account/getHistory/{account}")

public List<UserHistory> getHistory(@PathVariable("account") long account )

{

List<UserHistory> history=hservice.getHistory(account);

Collections.reverse(history);

return history;

}

@GetMapping("/account/getTransfers/{account}")

public List<Transfer> getTransfers(@PathVariable("account") long account )

{

return tservice.getTransfers(account);

}

}