```
package com.icinbank.exceptionHandling;

public class AccessDeniedException extends Exception {

    public AccessDeniedException() {
        super();
    }

    public AccessDeniedException(String message) {
        super(message);
    }
}
```

```
package com.icinbank.bean;

public class AccessUpdateBody {

    private String firstname;

    private String lastname;

    private boolean accessible;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public boolean isAccessible() {
        return accessible;
    }

    public void setAccessible(boolean accessible) {
        this.accessible = accessible;
    }
}
```

```
package com.icinbank.bean;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class Account {

    @Id
    private int id;

    private double balance;

    private int user_id;

    private boolean saving;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public int getUser_id() {
        return user_id;
    }

    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.*;
import com.icinbank.exceptionHandling.AccessDeniedException;
import com.icinbank.exceptionHandling.InsufficientFundException;
import com.icinbank.exceptionHandling.NotRecipientFoundException;
import com.icinbank.exceptionHandling.UserBlockedException;
import com.icinbank.repository.AccountRepository;
import com.icinbank.repository.TransactionRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.sql.Timestamp;
import java.util.Date;
import java.util.List;

@RestController
@RequestMapping("accounts")
public class AccountController {

    Date date = new Date();

    @Autowired
    AccountRepository accountRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    TransactionRepository transactionRepository;

    @GetMapping("balance/{id}")
    public double getBalance(@PathVariable("id") int id) {
        return accountRepository.getBalanceByAccountId(id);
    }

    @PostMapping("deposit")
    public void deposit(@RequestBody DepositRequestBody depositBody) throws AccessDeniedException,
UserBlockedException {
        int userId = depositBody.getUserId();
        User user = userRepository.getUserById(userId).get(0);
        if (user.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!user.isDepositAccess()) {
            throw new AccessDeniedException("Not allow to make deposit.");
        }
        int accountId;
        if (depositBody.isSaving()) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
```

```
        }
        double amount = depositBody.getAmount();
        double oldBalance = accountRepository.getBalanceByAccountId(accountId);
        double newBalance = oldBalance + amount;
        accountRepository.changeBalance(newBalance, accountId);
    }

    @PostMapping("withdrawal")
    public void withdrawal(@RequestBody WithdrawalRequestBody withdrawalRequestBody) throws
InsufficientFundException, AccessDeniedException, UserBlockedException {
        int userId = withdrawalRequestBody.getUserId();
        User user = userRepository.getUserById(userId).get(0);
        if (user.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!user.isWithdrawalAccess()) {
            throw new AccessDeniedException("Not allow to make withdrawal.");
        }
        int accountId;
        if (withdrawalRequestBody.isSaving()) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
        }
        double amount = withdrawalRequestBody.getAmount();
        double availableAmount = accountRepository.getBalanceByAccountId(accountId);
        if (availableAmount < amount) {
            throw new InsufficientFundException("Insufficient fund in account.");
        }
        double newAmount = availableAmount - amount;
        accountRepository.changeBalance(newAmount, accountId);
    }

    @PostMapping("transfer")
    public void makeTransfer(@RequestBody TransferRequestBody transferBody) throws NotRecipientFoundException,
InsufficientFundException, AccessDeniedException, UserBlockedException {
        int originId = transferBody.getUserIdFrom();
        User originUser = userRepository.getUserById(originId).get(0);
        if (originUser.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!originUser.isTransferAccess()) {
            throw new AccessDeniedException("Not allow to make transfer.");
        }
        String firstnameTo = transferBody.getFirstnameTo();
        String lastnameTo = transferBody.getLastnameTo();
        List<User> user = userRepository.getUserByFirstnameAndLastname(firstnameTo, lastnameTo);
        if (user == null || user.size() == 0) {
            throw new NotRecipientFoundException("Not recipient found.");
        }
        User recipient = user.get(0);
        double amount = transferBody.getAmount();
        User origin = userRepository.getUserById(transferBody.getUserIdFrom()).get(0);
        int accountIdFrom;
```

```
        int accountIdTo;
        if (transferBody.isFromSaving()) {
            accountIdFrom = Integer.parseInt(String.valueOf(origin.getId()) + "02");
        } else {
            accountIdFrom = Integer.parseInt(String.valueOf(origin.getId()) + "01");
        }
        double availableBalance = accountRepository.getBalanceByAccountId(accountIdFrom);
        if (availableBalance < amount) {
            throw new InsufficientFundException("Insufficient fund in account.");
        }

        if (transferBody.isToSaving()) {
            accountIdTo = Integer.parseInt(String.valueOf(recipient.getId()) + "02");
        } else {
            accountIdTo = Integer.parseInt(String.valueOf(recipient.getId()) + "01");
        }
        double origienBalanceAfter = availableBalance - amount;
        double recipientBalanceAfter = accountRepository.getBalanceByAccountId(accountIdTo) + amount;
        accountRepository.changeBalance(origienBalanceAfter, accountIdFrom);
        accountRepository.changeBalance(recipientBalanceAfter, accountIdTo);
        //transactionRepository.insertTransaction(accountIdFrom, origin.getId(), accountIdTo, recipient.getId(), amount,
new Timestamp(date.getTime()));
        Transaction newTransaction = getNewTransaction(accountIdFrom, origin.getId(), accountIdTo, recipient.getId(),
amount, new Timestamp(date.getTime()));
        transactionRepository.save(newTransaction);
    }

    private Transaction getNewTransaction(int accountIdFrom, int userIdFrom, int accountIdTo, int userIdTo, double
amount, Timestamp time) {
        Transaction newTransaction = new Transaction();
        newTransaction.setAccountIdFrom(accountIdFrom);
        newTransaction.setAccountIdTo(accountIdTo);
        newTransaction.setAmount(amount);
        newTransaction.setUserIdFrom(userIdFrom);
        newTransaction.setUserIdTo(userIdTo);
        newTransaction.setTime(time);
        return newTransaction;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.Account;

public interface AccountRepository extends CrudRepository<Account, Integer> {

    @Modifying
    @Transactional
    @Query(value = "INSERT INTO account (id, user_id, saving, balance) VALUES (:id, :user_id, :saving, :balance)",
nativeQuery = true)
    public int addNewAccount(@Param("id") int id, @Param("user_id") int user_id, @Param("saving") boolean saving,
@Param("balance") double balance);

    @Query(value = "SELECT balance FROM account WHERE id = :id", nativeQuery = true)
    public double getBalanceByAccountId(@Param("id") int id);

    @Modifying
    @Transactional
    @Query(value = "UPDATE account SET balance = :balance WHERE id = :id", nativeQuery = true)
    public int changeBalance(@Param("balance") double balance, @Param("id") int id);
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class AdminUser {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.AccessUpdateBody;
import com.icinbank.bean.AdminUser;
import com.icinbank.bean.User;
import com.icinbank.bean.UserPostBody;
import com.icinbank.exceptionHandling.LoginFailedException;
import com.icinbank.exceptionHandling.UserNotFoundException;
import com.icinbank.repository.AdminUserRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("admin")
public class AdminUserController {

    @Autowired
    AdminUserRepository adminUserRepository;

    @Autowired
    UserRepository userRepository;

    //Change Deposit access
    @PostMapping("updateDepositAccess")
    public void updateDepositAccess(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException
{
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateDepositAccess(userId, accessible);
    }

    //Change Withdrawal access
    @PostMapping("updateWithdrawalAccess")
    public void updateWithdrawalAccess(@RequestBody AccessUpdateBody updateBody) throws
UserNotFoundException {
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
```

```
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateWithdrawalAccess(userId, accessible);
    }

    //Change Transfer access
    @PostMapping("updateTransferAccess")
    public void updateTransferAccess(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException
{
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateTransferAccess(userId, accessible);
    }

    //Change Block Status
    @PostMapping("updateBlockStatus")
    public void updateBlockStatus(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException {
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateBlockStatus(userId, accessible);
    }

    //Login
    @PostMapping("login")
    public AdminUser login(@RequestBody UserPostBody userBody) throws UserNotFoundException,
LoginFailedException {
        String email = userBody.getEmail();
        String password = userBody.getPassword();

        List<AdminUser> users = adminUserRepository.getUserByEmail(email);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("User not found.");
        }
        AdminUser user = users.get(0);
        if (!user.getPassword().equals(password)) {
            throw new LoginFailedException("Incorrect password.");
        }
```

```
        return user;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.icinbank.bean.AdminUser;

import java.util.List;

public interface AdminUserRepository extends CrudRepository<AdminUser, Integer> {

    @Query(value = "SELECT * FROM admin_user WHERE username = :username", nativeQuery = true)
    public List<AdminUser> getUserByUsername(@Param("username") String username);

    @Query(value = "SELECT * FROM admin_user WHERE email = :email", nativeQuery = true)
    public List<AdminUser> getUserByEmail(@Param("email") String email);
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.ChequeBookRequest;

import java.util.List;

public interface ChequeBookRepository extends CrudRepository<ChequeBookRequest, Integer> {


    @Modifying
    @Transactional
    @Query(value = "UPDATE cheque_book_request SET status = :status WHERE id = :id", nativeQuery = true)
    public int approveChequeRequest(@Param("id") int id, @Param("status") String status);

    @Query(value = "SELECT * FROM cheque_book_request WHERE user_id = :userId", nativeQuery = true)
    public List<ChequeBookRequest> getRequestsByUserId(@Param("userId") int userId);

    //Get all pending request
    @Query(value = "SELECT * FROM cheque_book_request WHERE status = 'REQUESTED'", nativeQuery = true)
    public List<ChequeBookRequest> getAllPendingRequest();
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class ChequeBookRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private int userId;

    private int accountId;

    private int quantity;

    @Column(columnDefinition = "varchar(20) default 'REQUESTED'")
    private String status = "REQUESTED";

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
```

```
    public int getAccountId() {
        return accountId;
    }

    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.*;
import com.icinbank.repository.ChequeBookRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("cheque")
public class ChequeController {

    @Autowired
    ChequeBookRepository chequeBookRepository;

    @Autowired
    UserRepository userRepository;

    @GetMapping("allPending")
    public List<pendingChequeBookRequest> getAllPendingRequest() {
        List<ChequeBookRequest> requests = chequeBookRepository.getAllPendingRequest();
        List<pendingChequeBookRequest> result = new ArrayList<>();
        for (ChequeBookRequest request : requests) {
            int requestId = request.getId();
            int accountId = request.getAccountId();
            int quantity = request.getQuantity();
            int userId = request.getUserId();
            String account = accountId % 2 == 0 ? "saving" : "checking";
            String status = request.getStatus();
            User customer = userRepository.getUserById(userId).get(0);
            String name = customer.getFirstname() + " " + customer.getLastname();
            pendingChequeBookRequest pendingRequest = new pendingChequeBookRequest(requestId, name, account,
quantity, status);
            result.add(pendingRequest);
        }
        return result;
    }

    @GetMapping("get/{userId}")

    public List<ChequeBookRequest> getRequestsByUserId(@PathVariable("userId") int userId) {
        return chequeBookRepository.getRequestsByUserId(userId);
    }

    @PostMapping("approve")
    public void approveRequest(@RequestBody UpdateRequestBody updateRequestBody) {
        int id = updateRequestBody.getId();
        String status = updateRequestBody.getStatus();
        chequeBookRepository.approveChequeRequest(id, status);
    }
```

```
    @PostMapping("request")
    public void requestChequeBook(@RequestBody ChequeRequestBody chequeRequestBody) {
        int userId = chequeRequestBody.getUserId();
        int quantity = chequeRequestBody.getQuantity();
        boolean isSaving = chequeRequestBody.isSaving();
        int accountId;
        if (isSaving) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
        }
        ChequeBookRequest newChequeBookRequest = getNewChequeBookRequest(userId, quantity, accountId);
        chequeBookRepository.save(newChequeBookRequest);
    }

    private ChequeBookRequest getNewChequeBookRequest(int userId, int quantity, int accountId) {
        ChequeBookRequest newChequeBookRequest = new ChequeBookRequest();
        newChequeBookRequest.setUserId(userId);
        newChequeBookRequest.setQuantity(quantity);
        newChequeBookRequest.setAccountId(accountId);
        return newChequeBookRequest;
    }

}
```

```
package com.icinbank.bean;

public class ChequeRequestBody {

    private int userId;

    private int quantity;

    private boolean saving;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.bean;

public class DepositRequestBody {

    private int userId;

    private double amount;

    private boolean saving;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.exceptionHandling;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

import javax.servlet.http.HttpServletRequest;

@ControllerAdvice
public class ExceptionHandlerControllerAdvice {

    @ExceptionHandler(UserNotFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody
    ExceptionResponse handleUserNotFoundException(final UserNotFoundException exception,
                                                  final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(LoginFailedException.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody
    ExceptionResponse handleLoginFailedException(final LoginFailedException exception,
                                                 final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(InsufficientFundException.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody
    ExceptionResponse InsufficientFundException(final InsufficientFundException exception,
                                                final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(NotRecipientFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody
    ExceptionResponse NotRecipientFoundException(final NotRecipientFoundException exception,
```

```
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(AccessDeniedException.class)
    @ResponseStatus(value = HttpStatus.UNAUTHORIZED)
    public @ResponseBody
    ExceptionResponse AccessDeniedException(final AccessDeniedException exception,
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(UserBlockedException.class)
    @ResponseStatus(value = HttpStatus.UNAUTHORIZED)
    public @ResponseBody
    ExceptionResponse UserBlockedException(final UserBlockedException exception,
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }
}
```

```
package com.icinbank.exceptionHandling;

public class ExceptionResponse {

    private String errorMessage;
    private String requestedURI;

    public String getErrorMessage() {
        return errorMessage;
    }

    public void setErrorMessage(final String errorMessage) {
        this.errorMessage = errorMessage;
    }

    public String getRequestedURI() {
        return requestedURI;
    }

    public void callerURL(final String requestedURI) {
        this.requestedURI = requestedURI;
    }
}
```

```
package com.icinbank;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class IcinbankApplication {

	public static void main(String[] args) {
		SpringApplication.run(IcinbankApplication.class, args);
	}

}
```

```
package com.icinbank;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class IcinbankApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

```
package com.icinbank.exceptionHandling;

public class InsufficientFundException extends Exception {

    public InsufficientFundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public InsufficientFundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.exceptionHandling;

public class LoginFailedException extends Exception {

    public LoginFailedException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public LoginFailedException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class NewUserPostBody {

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

}

```
package com.icinbank.exceptionHandling;

public class NotRecipientFoundException extends Exception {

    public NotRecipientFoundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public NotRecipientFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class pendingChequeBookRequest {

    private int requestId;

    private String customerName;

    private String account;

    private int quantity;

    private String status;

    public pendingChequeBookRequest(int requestId, String customerName, String account, int quantity, String status) {
        this.requestId = requestId;
        this.customerName = customerName;
        this.account = account;
        this.quantity = quantity;
        this.status = status;
    }

    public int getRequestId() {
        return requestId;
    }

    public void setRequestId(int requestId) {
        this.requestId = requestId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getAccount() {
        return account;
    }

    public void setAccount(String account) {
        this.account = account;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
```

```
    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

```
package com.icinbank.config;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class SimpleCORSFilter implements Filter {

   private final Logger log = LoggerFactory.getLogger(SimpleCORSFilter.class);

   public SimpleCORSFilter() {
      log.info("SimpleCORSFilter init");
   }

   @Override
   public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
         throws IOException, ServletException {

      HttpServletRequest request = (HttpServletRequest) req;
      HttpServletResponse response = (HttpServletResponse) res;

      response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"));
      response.setHeader("Access-Control-Allow-Credentials", "true");
      response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");
      response.setHeader("Access-Control-Max-Age", "3600");
      response.setHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With, remember-
me");

      chain.doFilter(req, res);
   }

   @Override
   public void init(FilterConfig filterConfig) {
   }

   @Override
   public void destroy() {
   }
}
```

```
package com.icinbank.bean;

import javax.persistence.*;
import java.sql.Timestamp;

@Entity
@Table
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private int accountIdFrom;

    private int userIdFrom;

    private int accountIdTo;

    private int userIdTo;

    private double amount;

    private Timestamp time;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getAccountIdFrom() {
        return accountIdFrom;
    }

    public void setAccountIdFrom(int accountIdFrom) {
        this.accountIdFrom = accountIdFrom;
    }

    public int getUserIdFrom() {
        return userIdFrom;
    }

    public void setUserIdFrom(int userIdFrom) {
        this.userIdFrom = userIdFrom;
    }

    public int getAccountIdTo() {
        return accountIdTo;
    }
```

```
    public void setAccountIdTo(int accountIdTo) {
        this.accountIdTo = accountIdTo;
    }

    public int getUserIdTo() {
        return userIdTo;
    }

    public void setUserIdTo(int userIdTo) {
        this.userIdTo = userIdTo;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public Timestamp getTime() {
        return time;
    }

    public void setTime(Timestamp time) {
        this.time = time;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.Transaction;
import com.icinbank.bean.TransactionHistory;
import com.icinbank.bean.User;
import com.icinbank.repository.TransactionRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("transactions")
public class TransactionController {

    @Autowired
    TransactionRepository transactionRepository;

    @Autowired
    UserRepository userRepository;

    //Get transaction history
    @GetMapping("getHistory/{id}")
    public List<TransactionHistory> getTransactionHistory(@PathVariable("id") int id) {
        List<TransactionHistory> history = new ArrayList<>();
        List<Transaction> transactions = transactionRepository.getTransactions(id);

        for (Transaction transaction : transactions) {
            TransactionHistory th = new TransactionHistory();
            User fromUser = userRepository.getUserById(transaction.getUserIdFrom()).get(0);
            User toUser = userRepository.getUserById(transaction.getUserIdTo()).get(0);
            double amount = transaction.getAmount();
            int accountIdFrom = transaction.getAccountIdFrom();
            int accountIdTo = transaction.getAccountIdTo();
            String fromName = fromUser.getFirstname() + " " + fromUser.getLastname();
            String toName = toUser.getFirstname() + " " + toUser.getLastname();
            String fromAccount = accountIdFrom % 2 == 0 ? "saving" : "checking";
            String toAccount = accountIdTo % 2 == 0 ? "saving" : "checking";
            th.setAmount(amount);
            th.setFrom(fromName);
            th.setTo(toName);
            th.setFromAccount(fromAccount);
            th.setToAccount(toAccount);
            history.add(th);
        }
        return history;
    }
}
```

```
package com.icinbank.bean;

public class TransactionHistory {

    private String from;

    private String to;

    private String fromAccount;

    private String toAccount;

    private double amount;

    public String getFrom() {
        return from;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public String getTo() {
        return to;
    }

    public void setTo(String to) {
        this.to = to;
    }

    public String getFromAccount() {
        return fromAccount;
    }

    public void setFromAccount(String fromAccount) {
        this.fromAccount = fromAccount;
    }

    public String getToAccount() {
        return toAccount;
    }

    public void setToAccount(String toAccount) {
        this.toAccount = toAccount;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }
```

}

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.icinbank.bean.Transaction;

import java.sql.Timestamp;
import java.util.List;

public interface TransactionRepository extends CrudRepository<Transaction, Integer> {


    @Query(value = "INSERT INTO transaction (accountIdFrom, userIdFrom, accountIdTo, userIdTo, amount, time) " +
            "VALUES (:accountIdFrom, :userIdFrom, :accountIdTo, :userIdTo, :amount, :time)", nativeQuery = true)
    public List<Transaction> insertTransaction(@Param("accountIdFrom") int accountIdFrom, @Param("userIdFrom")
int userIdFrom,
                                               @Param("accountIdTo") int accountIdTo, @Param("userIdTo") int userIdTo,
                                               @Param("amount") double amount, @Param("time") Timestamp time);

    @Query(value = "SELECT * FROM transaction WHERE account_id_from = :accountId OR account_id_to =
:accountId", nativeQuery = true)
    public List<Transaction> getTransactions(@Param("accountId") int accountId);
}
```

```
package com.icinbank.bean;

public class TransferRequestBody {

    private int userIdFrom;

    private String firstnameTo;

    private String lastnameTo;

    private double amount;

    private boolean fromSaving;

    private boolean toSaving;

    public int getUserIdFrom() {
        return userIdFrom;
    }

    public void setUserIdFrom(int userIdFrom) {
        this.userIdFrom = userIdFrom;
    }

    public String getFirstnameTo() {
        return firstnameTo;
    }

    public void setFirstnameTo(String firstnameTo) {
        this.firstnameTo = firstnameTo;
    }

    public String getLastnameTo() {
        return lastnameTo;
    }

    public void setLastnameTo(String lastnameTo) {
        this.lastnameTo = lastnameTo;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public boolean isFromSaving() {
        return fromSaving;
    }

    public void setFromSaving(boolean fromSaving) {
```

```
        this.fromSaving = fromSaving;
    }

    public boolean isToSaving() {
        return toSaving;
    }

    public void setToSaving(boolean toSaving) {
        this.toSaving = toSaving;
    }
}
```

```
package com.icinbank.bean;

public class UpdateRequestBody {

    private int id;

    private String status;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    @Column(columnDefinition = "boolean default false")
    private boolean blocked;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean transferAccess = true;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean depositAccess = true;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean withdrawalAccess = true;


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
```

```
}

public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public boolean isBlocked() {
    return blocked;
}

public void setBlocked(boolean blocked) {
    this.blocked = blocked;
}

public boolean isTransferAccess() {
    return transferAccess;
}

public void setTransferAccess(boolean transferAccess) {
    this.transferAccess = transferAccess;
}

public boolean isDepositAccess() {
    return depositAccess;
}

public void setDepositAccess(boolean depositAccess) {
    this.depositAccess = depositAccess;
}
```

```
    public boolean isWithdrawalAccess() {
        return withdrawalAccess;
    }

    public void setWithdrawalAccess(boolean withdrawalAccess) {
        this.withdrawalAccess = withdrawalAccess;
    }
}
```

```
package com.icinbank.exceptionHandling;

public class UserBlockedException extends Exception {

    public UserBlockedException() {
        super();
    }

    public UserBlockedException(String message) {
        super(message);
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.User;
import com.icinbank.bean.UserPostBody;
import com.icinbank.exceptionHandling.LoginFailedException;
import com.icinbank.exceptionHandling.UserBlockedException;
import com.icinbank.exceptionHandling.UserNotFoundException;
import com.icinbank.repository.AccountRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("users")
public class UserController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    AccountRepository accountRepository;

    //Login
    @PostMapping("login")
    public User login(@RequestBody UserPostBody userBody) throws UserNotFoundException, LoginFailedException,
UserBlockedException {
        String email = userBody.getEmail();
        String password = userBody.getPassword();

        List<User> users = userRepository.getUserByEmail(email);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("User not found.");
        }
        User user = users.get(0);
        if (!user.getPassword().equals(password)) {
            throw new LoginFailedException("Incorrect password.");
        }
        if (user.isBlocked()) {
            throw new UserBlockedException("Your account is blocked!");
        }
        return user;
    }

    @PostMapping("add")
    public User addUser(@RequestBody User newUser) {
        User user = userRepository.save(newUser);
        int userId = user.getId();
        int checkingAccountId = Integer.parseInt(String.valueOf(userId) + "01");
```

```
        int savingAccountId = Integer.parseInt(String.valueOf(userId) + "02");
        accountRepository.addNewAccount(checkingAccountId, userId, false, 0.00);
        accountRepository.addNewAccount(savingAccountId, userId, true, 0.00);
        return user;
    }

}
```

```
package com.icinbank.exceptionHandling;

public class UserNotFoundException extends Exception {

    public UserNotFoundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public UserNotFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class UserPostBody {

    private String email;

    private String password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.User;

import java.util.List;

public interface UserRepository extends CrudRepository<User, Integer> {

    @Query(value = "SELECT * FROM user WHERE username = :username", nativeQuery = true)
    public List<User> getUserByUsername(@Param("username") String username);

    @Query(value = "SELECT * FROM user WHERE email = :email", nativeQuery = true)
    public List<User> getUserByEmail(@Param("email") String email);

    @Query(value = "SELECT * FROM user WHERE firstname = :firstname AND lastname = :lastname", nativeQuery
= true)
    public List<User> getUserByFirstnameAndLastname(@Param("firstname") String firstname, @Param("lastname")
String lastname);

    @Query(value = "SELECT * FROM user WHERE id = :id", nativeQuery = true)
    public List<User> getUserById(@Param("id") int id);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET deposit_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateDepositAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET withdrawal_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateWithdrawalAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET transfer_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateTransferAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET blocked = :accessible WHERE id = :id", nativeQuery = true)
    public int updateBlockStatus(@Param("id") int id, @Param("accessible") boolean accessible);
}
```

```
package com.icinbank.bean;

public class WithdrawalRequestBody {

    private int userId;

    private boolean saving;

    private double amount;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }
}
```

```
package com.icinbank.exceptionHandling;

public class AccessDeniedException extends Exception {

    public AccessDeniedException() {
        super();
    }

    public AccessDeniedException(String message) {
        super(message);
    }
}
```

```
package com.icinbank.bean;

public class AccessUpdateBody {

    private String firstname;

    private String lastname;

    private boolean accessible;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public boolean isAccessible() {
        return accessible;
    }

    public void setAccessible(boolean accessible) {
        this.accessible = accessible;
    }
}
```

```
package com.icinbank.bean;

import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table
public class Account {

    @Id
    private int id;

    private double balance;

    private int user_id;

    private boolean saving;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

    public int getUser_id() {
        return user_id;
    }

    public void setUser_id(int user_id) {
        this.user_id = user_id;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.*;
import com.icinbank.exceptionHandling.AccessDeniedException;
import com.icinbank.exceptionHandling.InsufficientFundException;
import com.icinbank.exceptionHandling.NotRecipientFoundException;
import com.icinbank.exceptionHandling.UserBlockedException;
import com.icinbank.repository.AccountRepository;
import com.icinbank.repository.TransactionRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.sql.Timestamp;
import java.util.Date;
import java.util.List;

@RestController
@RequestMapping("accounts")
public class AccountController {

    Date date = new Date();

    @Autowired
    AccountRepository accountRepository;

    @Autowired
    UserRepository userRepository;

    @Autowired
    TransactionRepository transactionRepository;

    @GetMapping("balance/{id}")
    public double getBalance(@PathVariable("id") int id) {
        return accountRepository.getBalanceByAccountId(id);
    }

    @PostMapping("deposit")
    public void deposit(@RequestBody DepositRequestBody depositBody) throws AccessDeniedException,
UserBlockedException {
        int userId = depositBody.getUserId();
        User user = userRepository.getUserById(userId).get(0);
        if (user.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!user.isDepositAccess()) {
            throw new AccessDeniedException("Not allow to make deposit.");
        }
        int accountId;
        if (depositBody.isSaving()) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
```

```
        }
        double amount = depositBody.getAmount();
        double oldBalance = accountRepository.getBalanceByAccountId(accountId);
        double newBalance = oldBalance + amount;
        accountRepository.changeBalance(newBalance, accountId);
    }

    @PostMapping("withdrawal")
    public void withdrawal(@RequestBody WithdrawalRequestBody withdrawalRequestBody) throws
InsufficientFundException, AccessDeniedException, UserBlockedException {
        int userId = withdrawalRequestBody.getUserId();
        User user = userRepository.getUserById(userId).get(0);
        if (user.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!user.isWithdrawalAccess()) {
            throw new AccessDeniedException("Not allow to make withdrawal.");
        }
        int accountId;
        if (withdrawalRequestBody.isSaving()) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
        }
        double amount = withdrawalRequestBody.getAmount();
        double availableAmount = accountRepository.getBalanceByAccountId(accountId);
        if (availableAmount < amount) {
            throw new InsufficientFundException("Insufficient fund in account.");
        }
        double newAmount = availableAmount - amount;
        accountRepository.changeBalance(newAmount, accountId);
    }

    @PostMapping("transfer")
    public void makeTransfer(@RequestBody TransferRequestBody transferBody) throws NotRecipientFoundException,
InsufficientFundException, AccessDeniedException, UserBlockedException {
        int originId = transferBody.getUserIdFrom();
        User originUser = userRepository.getUserById(originId).get(0);
        if (originUser.isBlocked()) {
            throw new UserBlockedException("User is blocked.");
        }
        if (!originUser.isTransferAccess()) {
            throw new AccessDeniedException("Not allow to make transfer.");
        }
        String firstnameTo = transferBody.getFirstnameTo();
        String lastnameTo = transferBody.getLastnameTo();
        List<User> user = userRepository.getUserByFirstnameAndLastname(firstnameTo, lastnameTo);
        if (user == null || user.size() == 0) {
            throw new NotRecipientFoundException("Not recipient found.");
        }
        User recipient = user.get(0);
        double amount = transferBody.getAmount();
        User origin = userRepository.getUserById(transferBody.getUserIdFrom()).get(0);
        int accountIdFrom;
```

```
        int accountIdTo;
        if (transferBody.isFromSaving()) {
            accountIdFrom = Integer.parseInt(String.valueOf(origin.getId()) + "02");
        } else {
            accountIdFrom = Integer.parseInt(String.valueOf(origin.getId()) + "01");
        }
        double availableBalance = accountRepository.getBalanceByAccountId(accountIdFrom);
        if (availableBalance < amount) {
            throw new InsufficientFundException("Insufficient fund in account.");
        }

        if (transferBody.isToSaving()) {
            accountIdTo = Integer.parseInt(String.valueOf(recipient.getId()) + "02");
        } else {
            accountIdTo = Integer.parseInt(String.valueOf(recipient.getId()) + "01");
        }
        double origienBalanceAfter = availableBalance - amount;
        double recipientBalanceAfter = accountRepository.getBalanceByAccountId(accountIdTo) + amount;
        accountRepository.changeBalance(origienBalanceAfter, accountIdFrom);
        accountRepository.changeBalance(recipientBalanceAfter, accountIdTo);
        //transactionRepository.insertTransaction(accountIdFrom, origin.getId(), accountIdTo, recipient.getId(), amount,
new Timestamp(date.getTime()));
        Transaction newTransaction = getNewTransaction(accountIdFrom, origin.getId(), accountIdTo, recipient.getId(),
amount, new Timestamp(date.getTime()));
        transactionRepository.save(newTransaction);
    }

    private Transaction getNewTransaction(int accountIdFrom, int userIdFrom, int accountIdTo, int userIdTo, double
amount, Timestamp time) {
        Transaction newTransaction = new Transaction();
        newTransaction.setAccountIdFrom(accountIdFrom);
        newTransaction.setAccountIdTo(accountIdTo);
        newTransaction.setAmount(amount);
        newTransaction.setUserIdFrom(userIdFrom);
        newTransaction.setUserIdTo(userIdTo);
        newTransaction.setTime(time);
        return newTransaction;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.Account;

public interface AccountRepository extends CrudRepository<Account, Integer> {

    @Modifying
    @Transactional
    @Query(value = "INSERT INTO account (id, user_id, saving, balance) VALUES (:id, :user_id, :saving, :balance)",
nativeQuery = true)
    public int addNewAccount(@Param("id") int id, @Param("user_id") int user_id, @Param("saving") boolean saving,
@Param("balance") double balance);

    @Query(value = "SELECT balance FROM account WHERE id = :id", nativeQuery = true)
    public double getBalanceByAccountId(@Param("id") int id);

    @Modifying
    @Transactional
    @Query(value = "UPDATE account SET balance = :balance WHERE id = :id", nativeQuery = true)
    public int changeBalance(@Param("balance") double balance, @Param("id") int id);
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class AdminUser {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.AccessUpdateBody;
import com.icinbank.bean.AdminUser;
import com.icinbank.bean.User;
import com.icinbank.bean.UserPostBody;
import com.icinbank.exceptionHandling.LoginFailedException;
import com.icinbank.exceptionHandling.UserNotFoundException;
import com.icinbank.repository.AdminUserRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("admin")
public class AdminUserController {

    @Autowired
    AdminUserRepository adminUserRepository;

    @Autowired
    UserRepository userRepository;

    //Change Deposit access
    @PostMapping("updateDepositAccess")
    public void updateDepositAccess(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException
{
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateDepositAccess(userId, accessible);
    }

    //Change Withdrawal access
    @PostMapping("updateWithdrawalAccess")
    public void updateWithdrawalAccess(@RequestBody AccessUpdateBody updateBody) throws
UserNotFoundException {
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
```

```
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateWithdrawalAccess(userId, accessible);
    }

    //Change Transfer access
    @PostMapping("updateTransferAccess")
    public void updateTransferAccess(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException
{
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateTransferAccess(userId, accessible);
    }

    //Change Block Status
    @PostMapping("updateBlockStatus")
    public void updateBlockStatus(@RequestBody AccessUpdateBody updateBody) throws UserNotFoundException {
        String firstname = updateBody.getFirstname();
        String lastname = updateBody.getLastname();
        List<User> users = userRepository.getUserByFirstnameAndLastname(firstname, lastname);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("Not such user found.");
        }
        User user = users.get(0);
        int userId = user.getId();
        boolean accessible = updateBody.isAccessible();
        userRepository.updateBlockStatus(userId, accessible);
    }

    //Login
    @PostMapping("login")
    public AdminUser login(@RequestBody UserPostBody userBody) throws UserNotFoundException,
LoginFailedException {
        String email = userBody.getEmail();
        String password = userBody.getPassword();

        List<AdminUser> users = adminUserRepository.getUserByEmail(email);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("User not found.");
        }
        AdminUser user = users.get(0);
        if (!user.getPassword().equals(password)) {
            throw new LoginFailedException("Incorrect password.");
        }
```

```
        return user;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.icinbank.bean.AdminUser;

import java.util.List;

public interface AdminUserRepository extends CrudRepository<AdminUser, Integer> {

    @Query(value = "SELECT * FROM admin_user WHERE username = :username", nativeQuery = true)
    public List<AdminUser> getUserByUsername(@Param("username") String username);

    @Query(value = "SELECT * FROM admin_user WHERE email = :email", nativeQuery = true)
    public List<AdminUser> getUserByEmail(@Param("email") String email);
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.ChequeBookRequest;

import java.util.List;

public interface ChequeBookRepository extends CrudRepository<ChequeBookRequest, Integer> {


    @Modifying
    @Transactional
    @Query(value = "UPDATE cheque_book_request SET status = :status WHERE id = :id", nativeQuery = true)
    public int approveChequeRequest(@Param("id") int id, @Param("status") String status);

    @Query(value = "SELECT * FROM cheque_book_request WHERE user_id = :userId", nativeQuery = true)
    public List<ChequeBookRequest> getRequestsByUserId(@Param("userId") int userId);

    //Get all pending request
    @Query(value = "SELECT * FROM cheque_book_request WHERE status = 'REQUESTED'", nativeQuery = true)
    public List<ChequeBookRequest> getAllPendingRequest();
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class ChequeBookRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private int userId;

    private int accountId;

    private int quantity;

    @Column(columnDefinition = "varchar(20) default 'REQUESTED'")
    private String status = "REQUESTED";

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
```

```
    public int getAccountId() {
        return accountId;
    }

    public void setAccountId(int accountId) {
        this.accountId = accountId;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.*;
import com.icinbank.repository.ChequeBookRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("cheque")
public class ChequeController {

    @Autowired
    ChequeBookRepository chequeBookRepository;

    @Autowired
    UserRepository userRepository;

    @GetMapping("allPending")
    public List<pendingChequeBookRequest> getAllPendingRequest() {
        List<ChequeBookRequest> requests = chequeBookRepository.getAllPendingRequest();
        List<pendingChequeBookRequest> result = new ArrayList<>();
        for (ChequeBookRequest request : requests) {
            int requestId = request.getId();
            int accountId = request.getAccountId();
            int quantity = request.getQuantity();
            int userId = request.getUserId();
            String account = accountId % 2 == 0 ? "saving" : "checking";
            String status = request.getStatus();
            User customer = userRepository.getUserById(userId).get(0);
            String name = customer.getFirstname() + " " + customer.getLastname();
            pendingChequeBookRequest pendingRequest = new pendingChequeBookRequest(requestId, name, account,
quantity, status);
            result.add(pendingRequest);
        }
        return result;
    }

    @GetMapping("get/{userId}")

    public List<ChequeBookRequest> getRequestsByUserId(@PathVariable("userId") int userId) {
        return chequeBookRepository.getRequestsByUserId(userId);
    }

    @PostMapping("approve")
    public void approveRequest(@RequestBody UpdateRequestBody updateRequestBody) {
        int id = updateRequestBody.getId();
        String status = updateRequestBody.getStatus();
        chequeBookRepository.approveChequeRequest(id, status);
    }
```

```
    @PostMapping("request")
    public void requestChequeBook(@RequestBody ChequeRequestBody chequeRequestBody) {
        int userId = chequeRequestBody.getUserId();
        int quantity = chequeRequestBody.getQuantity();
        boolean isSaving = chequeRequestBody.isSaving();
        int accountId;
        if (isSaving) {
            accountId = Integer.parseInt(String.valueOf(userId) + "02");
        } else {
            accountId = Integer.parseInt(String.valueOf(userId) + "01");
        }
        ChequeBookRequest newChequeBookRequest = getNewChequeBookRequest(userId, quantity, accountId);
        chequeBookRepository.save(newChequeBookRequest);
    }

    private ChequeBookRequest getNewChequeBookRequest(int userId, int quantity, int accountId) {
        ChequeBookRequest newChequeBookRequest = new ChequeBookRequest();
        newChequeBookRequest.setUserId(userId);
        newChequeBookRequest.setQuantity(quantity);
        newChequeBookRequest.setAccountId(accountId);
        return newChequeBookRequest;
    }

}
```

```
package com.icinbank.bean;

public class ChequeRequestBody {

    private int userId;

    private int quantity;

    private boolean saving;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.bean;

public class DepositRequestBody {

    private int userId;

    private double amount;

    private boolean saving;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }
}
```

```
package com.icinbank.exceptionHandling;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ControllerAdvice;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.ResponseBody;
import org.springframework.web.bind.annotation.ResponseStatus;

import javax.servlet.http.HttpServletRequest;

@ControllerAdvice
public class ExceptionHandlerControllerAdvice {

    @ExceptionHandler(UserNotFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody
    ExceptionResponse handleUserNotFoundException(final UserNotFoundException exception,
                                                  final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(LoginFailedException.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody
    ExceptionResponse handleLoginFailedException(final LoginFailedException exception,
                                                 final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(InsufficientFundException.class)
    @ResponseStatus(value = HttpStatus.INTERNAL_SERVER_ERROR)
    public @ResponseBody
    ExceptionResponse InsufficientFundException(final InsufficientFundException exception,
                                                final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(NotRecipientFoundException.class)
    @ResponseStatus(value = HttpStatus.NOT_FOUND)
    public @ResponseBody
    ExceptionResponse NotRecipientFoundException(final NotRecipientFoundException exception,
```

```
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(AccessDeniedException.class)
    @ResponseStatus(value = HttpStatus.UNAUTHORIZED)
    public @ResponseBody
    ExceptionResponse AccessDeniedException(final AccessDeniedException exception,
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }

    @ExceptionHandler(UserBlockedException.class)
    @ResponseStatus(value = HttpStatus.UNAUTHORIZED)
    public @ResponseBody
    ExceptionResponse UserBlockedException(final UserBlockedException exception,
                                    final HttpServletRequest request) {
        ExceptionResponse error = new ExceptionResponse();
        error.setErrorMessage(exception.getMessage());
        error.callerURL(request.getRequestURI());

        return error;
    }
}
```

```
package com.icinbank.exceptionHandling;

public class ExceptionResponse {

    private String errorMessage;
    private String requestedURI;

    public String getErrorMessage() {
        return errorMessage;
    }

    public void setErrorMessage(final String errorMessage) {
        this.errorMessage = errorMessage;
    }

    public String getRequestedURI() {
        return requestedURI;
    }

    public void callerURL(final String requestedURI) {
        this.requestedURI = requestedURI;
    }
}
```

```
package com.icinbank;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class IcinbankApplication {

	public static void main(String[] args) {
		SpringApplication.run(IcinbankApplication.class, args);
	}

}
```

```
package com.icinbank;

import org.junit.jupiter.api.Test;
import org.springframework.boot.test.context.SpringBootTest;

@SpringBootTest
class IcinbankApplicationTests {

	@Test
	void contextLoads() {
	}

}
```

```
package com.icinbank.exceptionHandling;

public class InsufficientFundException extends Exception {

    public InsufficientFundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public InsufficientFundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.exceptionHandling;

public class LoginFailedException extends Exception {

    public LoginFailedException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public LoginFailedException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class NewUserPostBody {

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
```

}

```
package com.icinbank.exceptionHandling;

public class NotRecipientFoundException extends Exception {

    public NotRecipientFoundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public NotRecipientFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class pendingChequeBookRequest {

    private int requestId;

    private String customerName;

    private String account;

    private int quantity;

    private String status;

    public pendingChequeBookRequest(int requestId, String customerName, String account, int quantity, String status) {
        this.requestId = requestId;
        this.customerName = customerName;
        this.account = account;
        this.quantity = quantity;
        this.status = status;
    }

    public int getRequestId() {
        return requestId;
    }

    public void setRequestId(int requestId) {
        this.requestId = requestId;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getAccount() {
        return account;
    }

    public void setAccount(String account) {
        this.account = account;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
```

```
    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

```
package com.icinbank.config;

import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

import javax.servlet.*;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

@Component
public class SimpleCORSFilter implements Filter {

   private final Logger log = LoggerFactory.getLogger(SimpleCORSFilter.class);

   public SimpleCORSFilter() {
      log.info("SimpleCORSFilter init");
   }

   @Override
   public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
         throws IOException, ServletException {

      HttpServletRequest request = (HttpServletRequest) req;
      HttpServletResponse response = (HttpServletResponse) res;

      response.setHeader("Access-Control-Allow-Origin", request.getHeader("Origin"));
      response.setHeader("Access-Control-Allow-Credentials", "true");
      response.setHeader("Access-Control-Allow-Methods", "POST, PUT, GET, OPTIONS, DELETE");
      response.setHeader("Access-Control-Max-Age", "3600");
      response.setHeader("Access-Control-Allow-Headers", "Content-Type, Accept, X-Requested-With, remember-
me");

      chain.doFilter(req, res);
   }

   @Override
   public void init(FilterConfig filterConfig) {
   }

   @Override
   public void destroy() {
   }
}
```

```
package com.icinbank.bean;

import javax.persistence.*;
import java.sql.Timestamp;

@Entity
@Table
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private int accountIdFrom;

    private int userIdFrom;

    private int accountIdTo;

    private int userIdTo;

    private double amount;

    private Timestamp time;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getAccountIdFrom() {
        return accountIdFrom;
    }

    public void setAccountIdFrom(int accountIdFrom) {
        this.accountIdFrom = accountIdFrom;
    }

    public int getUserIdFrom() {
        return userIdFrom;
    }

    public void setUserIdFrom(int userIdFrom) {
        this.userIdFrom = userIdFrom;
    }

    public int getAccountIdTo() {
        return accountIdTo;
    }
```

```
    public void setAccountIdTo(int accountIdTo) {
        this.accountIdTo = accountIdTo;
    }

    public int getUserIdTo() {
        return userIdTo;
    }

    public void setUserIdTo(int userIdTo) {
        this.userIdTo = userIdTo;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public Timestamp getTime() {
        return time;
    }

    public void setTime(Timestamp time) {
        this.time = time;
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.Transaction;
import com.icinbank.bean.TransactionHistory;
import com.icinbank.bean.User;
import com.icinbank.repository.TransactionRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("transactions")
public class TransactionController {

    @Autowired
    TransactionRepository transactionRepository;

    @Autowired
    UserRepository userRepository;

    //Get transaction history
    @GetMapping("getHistory/{id}")
    public List<TransactionHistory> getTransactionHistory(@PathVariable("id") int id) {
        List<TransactionHistory> history = new ArrayList<>();
        List<Transaction> transactions = transactionRepository.getTransactions(id);

        for (Transaction transaction : transactions) {
            TransactionHistory th = new TransactionHistory();
            User fromUser = userRepository.getUserById(transaction.getUserIdFrom()).get(0);
            User toUser = userRepository.getUserById(transaction.getUserIdTo()).get(0);
            double amount = transaction.getAmount();
            int accountIdFrom = transaction.getAccountIdFrom();
            int accountIdTo = transaction.getAccountIdTo();
            String fromName = fromUser.getFirstname() + " " + fromUser.getLastname();
            String toName = toUser.getFirstname() + " " + toUser.getLastname();
            String fromAccount = accountIdFrom % 2 == 0 ? "saving" : "checking";
            String toAccount = accountIdTo % 2 == 0 ? "saving" : "checking";
            th.setAmount(amount);
            th.setFrom(fromName);
            th.setTo(toName);
            th.setFromAccount(fromAccount);
            th.setToAccount(toAccount);
            history.add(th);
        }
        return history;
    }
}
```

```
package com.icinbank.bean;

public class TransactionHistory {

    private String from;

    private String to;

    private String fromAccount;

    private String toAccount;

    private double amount;

    public String getFrom() {
        return from;
    }

    public void setFrom(String from) {
        this.from = from;
    }

    public String getTo() {
        return to;
    }

    public void setTo(String to) {
        this.to = to;
    }

    public String getFromAccount() {
        return fromAccount;
    }

    public void setFromAccount(String fromAccount) {
        this.fromAccount = fromAccount;
    }

    public String getToAccount() {
        return toAccount;
    }

    public void setToAccount(String toAccount) {
        this.toAccount = toAccount;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }
```

}

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;

import com.icinbank.bean.Transaction;

import java.sql.Timestamp;
import java.util.List;

public interface TransactionRepository extends CrudRepository<Transaction, Integer> {


    @Query(value = "INSERT INTO transaction (accountIdFrom, userIdFrom, accountIdTo, userIdTo, amount, time) " +
            "VALUES (:accountIdFrom, :userIdFrom, :accountIdTo, :userIdTo, :amount, :time)", nativeQuery = true)
    public List<Transaction> insertTransaction(@Param("accountIdFrom") int accountIdFrom, @Param("userIdFrom")
int userIdFrom,
                                               @Param("accountIdTo") int accountIdTo, @Param("userIdTo") int userIdTo,
                                               @Param("amount") double amount, @Param("time") Timestamp time);

    @Query(value = "SELECT * FROM transaction WHERE account_id_from = :accountId OR account_id_to =
:accountId", nativeQuery = true)
    public List<Transaction> getTransactions(@Param("accountId") int accountId);
}
```

```
package com.icinbank.bean;

public class TransferRequestBody {

    private int userIdFrom;

    private String firstnameTo;

    private String lastnameTo;

    private double amount;

    private boolean fromSaving;

    private boolean toSaving;

    public int getUserIdFrom() {
        return userIdFrom;
    }

    public void setUserIdFrom(int userIdFrom) {
        this.userIdFrom = userIdFrom;
    }

    public String getFirstnameTo() {
        return firstnameTo;
    }

    public void setFirstnameTo(String firstnameTo) {
        this.firstnameTo = firstnameTo;
    }

    public String getLastnameTo() {
        return lastnameTo;
    }

    public void setLastnameTo(String lastnameTo) {
        this.lastnameTo = lastnameTo;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }

    public boolean isFromSaving() {
        return fromSaving;
    }

    public void setFromSaving(boolean fromSaving) {
```

```
        this.fromSaving = fromSaving;
    }

    public boolean isToSaving() {
        return toSaving;
    }

    public void setToSaving(boolean toSaving) {
        this.toSaving = toSaving;
    }
}
```

```
package com.icinbank.bean;

public class UpdateRequestBody {

    private int id;

    private String status;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
        this.status = status;
    }
}
```

```
package com.icinbank.bean;

import javax.persistence.*;

@Entity
@Table
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String firstname;

    private String lastname;

    private String email;

    private String username;

    private String password;

    @Column(columnDefinition = "boolean default false")
    private boolean blocked;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean transferAccess = true;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean depositAccess = true;

    @Column(columnDefinition = "tinyint(1) default 1")
    private boolean withdrawalAccess = true;


    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getFirstname() {
        return firstname;
    }

    public void setFirstname(String firstname) {
        this.firstname = firstname;
    }

    public String getLastname() {
        return lastname;
```

```
}

public void setLastname(String lastname) {
    this.lastname = lastname;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getUsername() {
    return username;
}

public void setUsername(String username) {
    this.username = username;
}

public String getPassword() {
    return password;
}

public void setPassword(String password) {
    this.password = password;
}

public boolean isBlocked() {
    return blocked;
}

public void setBlocked(boolean blocked) {
    this.blocked = blocked;
}

public boolean isTransferAccess() {
    return transferAccess;
}

public void setTransferAccess(boolean transferAccess) {
    this.transferAccess = transferAccess;
}

public boolean isDepositAccess() {
    return depositAccess;
}

public void setDepositAccess(boolean depositAccess) {
    this.depositAccess = depositAccess;
}
```

```
    public boolean isWithdrawalAccess() {
        return withdrawalAccess;
    }

    public void setWithdrawalAccess(boolean withdrawalAccess) {
        this.withdrawalAccess = withdrawalAccess;
    }
}
```

```
package com.icinbank.exceptionHandling;

public class UserBlockedException extends Exception {

    public UserBlockedException() {
        super();
    }

    public UserBlockedException(String message) {
        super(message);
    }
}
```

```
package com.icinbank.controller;

import com.icinbank.bean.User;
import com.icinbank.bean.UserPostBody;
import com.icinbank.exceptionHandling.LoginFailedException;
import com.icinbank.exceptionHandling.UserBlockedException;
import com.icinbank.exceptionHandling.UserNotFoundException;
import com.icinbank.repository.AccountRepository;
import com.icinbank.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("users")
public class UserController {

    @Autowired
    UserRepository userRepository;

    @Autowired
    AccountRepository accountRepository;

    //Login
    @PostMapping("login")
    public User login(@RequestBody UserPostBody userBody) throws UserNotFoundException, LoginFailedException,
UserBlockedException {
        String email = userBody.getEmail();
        String password = userBody.getPassword();

        List<User> users = userRepository.getUserByEmail(email);
        if (users == null || users.size() == 0) {
            throw new UserNotFoundException("User not found.");
        }
        User user = users.get(0);
        if (!user.getPassword().equals(password)) {
            throw new LoginFailedException("Incorrect password.");
        }
        if (user.isBlocked()) {
            throw new UserBlockedException("Your account is blocked!");
        }
        return user;
    }

    @PostMapping("add")
    public User addUser(@RequestBody User newUser) {
        User user = userRepository.save(newUser);
        int userId = user.getId();
        int checkingAccountId = Integer.parseInt(String.valueOf(userId) + "01");
```

```
        int savingAccountId = Integer.parseInt(String.valueOf(userId) + "02");
        accountRepository.addNewAccount(checkingAccountId, userId, false, 0.00);
        accountRepository.addNewAccount(savingAccountId, userId, true, 0.00);
        return user;
    }

}
```

```
package com.icinbank.exceptionHandling;

public class UserNotFoundException extends Exception {

    public UserNotFoundException() {
        super();
        // TODO Auto-generated constructor stub
    }

    public UserNotFoundException(String message) {
        super(message);
        // TODO Auto-generated constructor stub
    }
}
```

```
package com.icinbank.bean;

public class UserPostBody {

    private String email;

    private String password;

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
}
```

```
package com.icinbank.repository;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.transaction.annotation.Transactional;

import com.icinbank.bean.User;

import java.util.List;

public interface UserRepository extends CrudRepository<User, Integer> {

    @Query(value = "SELECT * FROM user WHERE username = :username", nativeQuery = true)
    public List<User> getUserByUsername(@Param("username") String username);

    @Query(value = "SELECT * FROM user WHERE email = :email", nativeQuery = true)
    public List<User> getUserByEmail(@Param("email") String email);

    @Query(value = "SELECT * FROM user WHERE firstname = :firstname AND lastname = :lastname", nativeQuery
= true)
    public List<User> getUserByFirstnameAndLastname(@Param("firstname") String firstname, @Param("lastname")
String lastname);

    @Query(value = "SELECT * FROM user WHERE id = :id", nativeQuery = true)
    public List<User> getUserById(@Param("id") int id);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET deposit_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateDepositAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET withdrawal_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateWithdrawalAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET transfer_access = :accessible WHERE id = :id", nativeQuery = true)
    public int updateTransferAccess(@Param("id") int id, @Param("accessible") boolean accessible);

    @Modifying
    @Transactional
    @Query(value = "UPDATE user SET blocked = :accessible WHERE id = :id", nativeQuery = true)
    public int updateBlockStatus(@Param("id") int id, @Param("accessible") boolean accessible);
}
```

```
package com.icinbank.bean;

public class WithdrawalRequestBody {

    private int userId;

    private boolean saving;

    private double amount;

    public int getUserId() {
        return userId;
    }

    public void setUserId(int userId) {
        this.userId = userId;
    }

    public boolean isSaving() {
        return saving;
    }

    public void setSaving(boolean saving) {
        this.saving = saving;
    }

    public double getAmount() {
        return amount;
    }

    public void setAmount(double amount) {
        this.amount = amount;
    }
}
```