ICIN Bank

Application name: ICIN Bank
Developed by: Rajya Lakshmi Suvarna
HimaValli G

```java
package com.icin;

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);
        System.out.println("Server Started...");
    }

}
```

```java
package com.icin.controller;

import java.util.List;

import org.apache.commons.mail.EmailException;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import com.icin.entity.ChequebookRequest;
import com.icin.entity.User;
import com.icin.entity.UserDisplay;
import com.icin.service.AdminService;
import com.icin.service.MailService;

@RestController
@CrossOrigin(origins = "*", allowedHeaders = "*")
public class AdminController {

    @Autowired
    private AdminService service;

    @Autowired
    private MailService mailService;

    @GetMapping("user/{username}/features/{featureId}")
    public void setUserFeatures(
            @PathVariable("username") String username,
            @PathVariable("featureId") int featureId
            ) {
```

```
        service.setUserFeatures(username, featureId);
    }

    @GetMapping("user/{username}/authorize")
    public void authorizeUser(@PathVariable("username") String username) {
        try {
            service.authorizeUser(username);
            mailService.sendAuthorizedEmail(username);
        } catch (EmailException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    @GetMapping("user/{username}/authorize/cancel")
    public void cancelAuthorization(@PathVariable("username") String username) {
        try {
            mailService.sendAuthorizeCancelEmail(username);
        } catch (EmailException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        service.cancelAuthorization(username);
    }

    @GetMapping("user/unauthorized/all")
    public List<User> getAllUnauthorizedUsers() {
        return service.getAllUnauthorizedUsers();
    }

    @GetMapping("/user/all")
    public List<UserDisplay> getAllUsers() {
        return service.getAllUsers();
    }

    @GetMapping("/chequebook/request/all")
    public List<ChequebookRequest> getAllChequeBookRequests() {
        return service.getAllChequebookRequests();
    }

    @GetMapping("/user/{accNo}/chequebook/request/confirm")
    public void confirmChequeBookRequest(@PathVariable("accNo") long accNo) {
        service.acceptChequebookRequest(accNo);
    }

    @GetMapping("/user/{username}/enable")
    public void enableUser(@PathVariable("username") String username) {
        service.enableUser(username);
    }

    @GetMapping("/user/{username}/disable")
    public void disableUser(@PathVariable("username") String username) {
        service.disableUser(username);
    }

    @GetMapping("search/user/{userDetail}")
    public UserDisplay searchUser(@PathVariable("userDetail") String userDetail) {
        return service.searchUser(userDetail);
    }
```

```
}
```

```
package com.icin.controller;

import java.util.Collections;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.icin.details.TransactionDetails;
import com.icin.details.TransferDetails;
import com.icin.entity.Account;
import com.icin.entity.Saccount;
import com.icin.entity.Transfer;
import com.icin.entity.UserHistory;
import com.icin.repository.AccountRepository;
import com.icin.repository.SaccountRepository;
import com.icin.response.DepositResponse;
import com.icin.response.TransferResponse;
import com.icin.response.WithdrawResponse;
import com.icin.service.AccountsService;
import com.icin.service.SavingsAccountService;
import com.icin.service.TransferHistoryService;
import com.icin.service.UserHistoryService;

@RestController
@CrossOrigin(origins = "*", allowedHeaders = "*")
public class AccountController {

    @Autowired
    private AccountsService aservice;

    @Autowired
    private SavingsAccountService sservice;

    @Autowired
    private UserHistoryService hservice;

    @Autowired
    private TransferHistoryService tservice;

    @Autowired
    private AccountRepository aRepository;

    @Autowired
    private SaccountRepository sRepository;

    private final String ifsc = "ICIN7465";
```

```
    public AccountController() {
    }

    public static boolean isprimary(long account) {
        String s = Long.toString(account).substring(0, 10);
        String check = "3914918201";
        return s.equals(check);
    }

    @GetMapping({ "/account/details/{account}" })
    public Account getAccountDetails(@PathVariable("account") int account) {
        return this.aservice.getAccountDetails((long) account);
    }

    @PutMapping({ "/account/profile" })
    public Account updateProfile(@RequestBody Account account) {
        return this.aservice.updateAccount(account);
    }

    @GetMapping({ "/account/getprimary/{username}" })
    public Account getPrimarydetails(@PathVariable("username") String username) {
        return this.aservice.getAccount(username);
    }

    @GetMapping({ "/account/getsaving/{username}" })
    public Saccount getSavingdetails(@PathVariable("username") String username) {
        return this.sservice.getAccount(username);
    }

    @PostMapping({ "/account/deposit" })
    public DepositResponse deposit(@RequestBody TransactionDetails details) {
        return isprimary(details.getAccount()) ? this.aservice.deposit(details.getAccount(),
details.getAmount())
                : this.sservice.deposit(details.getAccount(), details.getAmount());
    }

    @PostMapping({ "/account/withdraw" })
    public WithdrawResponse withdraw(@RequestBody TransactionDetails details) {
        return isprimary(details.getAccount()) ? this.aservice.withdraw(details.getAccount(),
details.getAmount())
                : this.sservice.withdraw(details.getAccount(), details.getAmount());
    }

    @PostMapping({ "/account/transfer" })
    public TransferResponse transfer(@RequestBody TransferDetails details) {
        try {
            if (details.getIfsc().equals("ICIN7465")) {
                Account p = this.aRepository.findByUsername(details.getUsername());
                Saccount s = this.sRepository.findByUsername(details.getUsername());
                if (p.getAccno() != details.getSaccount() && s.getAccno() != details.getSaccount()) {
                    TransferResponse response = new TransferResponse();
                    response.setSaccount(details.getSaccount());
                    response.setResponseMessage(
                            "Dear user You can only transfer funds from the accounts registed with you");
                    response.setTransferStatus(false);
                    return response;
                } else {
                    return isprimary(details.getSaccount())
```

```
                            ? this.aservice.transfer(details.getSaccount(), details.getRaccount(),
details.getAmount())
                            : this.sservice.transfer(details.getSaccount(), details.getRaccount(),
details.getAmount());
                }
            } else {
                TransferResponse response = new TransferResponse();
                response.setSaccount(details.getSaccount());
                response.setResponseMessage("IFSC code is incorrect");
                response.setTransferStatus(false);
                return response;
            }
        } catch (Exception var5) {
            TransferResponse response = new TransferResponse();
            response.setSaccount(details.getSaccount());
            response.setResponseMessage("Please provide an IFSC code");
            response.setTransferStatus(false);
            return response;
        }
    }

    @GetMapping({ "/account/getHistory/{account}" })
    public List<UserHistory> getHistory(@PathVariable("account") long account) {
        List<UserHistory> history = this.hservice.getHistory(account);
        Collections.reverse(history);
        return history;
    }

    @GetMapping({ "/account/getTransfers/{account}" })
    public List<Transfer> getTransfers(@PathVariable("account") long account) {
        return this.tservice.getTransfers(account);
    }

}
```

```
package com.icin.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.icin.entity.ChequebookRequest;
import com.icin.response.ChequeResponse;
import com.icin.service.ChequebookService;

@RestController
@CrossOrigin(origins = { "*" }, allowedHeaders = { "*" })
public class ChequeBookController {

    @Autowired
    private ChequebookService service;
```

```
    public ChequeBookController() {
    }

    @PostMapping({ "/cheque/request" })
    public ChequeResponse createrequest(@RequestBody ChequebookRequest chequebook) {
        return this.service.createrequest(chequebook);
    }

    @GetMapping({ "/cheque/getbyAccount/{account}" })
    public List<ChequebookRequest> getRequests(@PathVariable("account") long account) {
        List<ChequebookRequest> list = this.service.getRequests(account);
        return list;
    }

}
```

```
package com.icin.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.icin.details.LoginDetails;
import com.icin.response.LoginResponse;
import com.icin.service.LoginService;

@RestController
@CrossOrigin(origins = { "*" }, allowedHeaders = { "*" })
public class LoginController {

    @Autowired
    LoginService service;

    public LoginController() {
    }

    @PostMapping({ "/login" })
    public LoginResponse userLogin(@RequestBody LoginDetails details) {
        return this.service.customerLogin(details);
    }

}
```

```
package com.icin.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
```

```java
import com.icin.details.UpdateDetails;
import com.icin.entity.User;
import com.icin.entity.UserDisplay;
import com.icin.response.UpdateResponse;
import com.icin.service.ProfileService;

@RestController
@CrossOrigin(origins = { "*" }, allowedHeaders = { "*" })
public class ProfileController {

    @Autowired
    private ProfileService pservice;

    public ProfileController() {
    }

    @PutMapping({ "/profile/update" })
    public UpdateResponse updateUser(@RequestBody UpdateDetails user) {
        return this.pservice.updateUser(user);
    }

    @GetMapping({ "/profile/{username}" })
    public User getUser(@PathVariable("username") String username) {
        return this.pservice.getUser(username);
    }

    @GetMapping({ "home/{username}" })
    public UserDisplay userDisplay(@PathVariable("username") String username) {
        return this.pservice.userDisplay(username);
    }

}
```

```java
package com.icin.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.icin.entity.User;
import com.icin.response.RegisterResponse;
import com.icin.service.RegistrationService;

@RestController
@CrossOrigin(origins = "*", allowedHeaders = "*")
public class RegistrationController {

    @Autowired
    private RegistrationService service;

    public RegistrationController() {
    }

    @PostMapping({"/register" })
```

```
    public RegisterResponse createUser(@RequestBody User user) {
        return this.service.createAccount(user);
    }

}
```

```
package com.icin.details;

public class LoginDetails {

    private String username;
    private String password;

    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.icin.details;

public class TransactionDetails {

    private long account;
    private int amount;


    public long getAccount() {
        return account;
    }
    public void setAccount(long account) {
        this.account = account;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }

}
```

```
package com.icin.details;
```

```java
public class TransferDetails {

    private long saccount;
    private long raccount;
    private int amount;
    private String Username;
    private String ifsc;

    public String getIfsc() {
        return ifsc;
    }

    public void setIfsc(String ifsc) {
        this.ifsc = ifsc;
    }

    public String getUsername() {
        return Username;
    }

    public void setUsername(String username) {
        Username = username;
    }

    public long getSaccount() {
        return saccount;
    }

    public void setSaccount(long saccount) {
        this.saccount = saccount;
    }

    public long getRaccount() {
        return raccount;
    }

    public void setRaccount(long raccount) {
        this.raccount = raccount;
    }

    public int getAmount() {
        return amount;
    }

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

}
```

```java
package com.icin.details;

public class UpdateDetails {

    private String username;
    private String password;
    private String newpassword;
```

```
    private long phone;
    private String address;
    private String 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 String getNewpassword() {
        return newpassword;
    }
    public void setNewpassword(String newpassword) {
        this.newpassword = newpassword;
    }
    public long getPhone() {
        return phone;
    }
    public void setPhone(long phone) {
        this.phone = phone;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }

    public UpdateDetails() {
        // TODO Auto-generated constructor stub
    }

}
```

```
package com.icin.entity;

import jakarta.persistence.*;

@Entity
@Table(name="account")
public class Account {


    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
```

```
    @Column(name = "id")
    private int id;

    private long accno;

    private int balance;

    private String username;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id")
    private User user;

    public String getUsername() {
        return username;
    }

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

    public int getId() {
        return id;
    }

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

    public long getAccno() {
        return accno;
    }

    public void setAccno(long accno) {
        this.accno = accno;
    }

    public int getBalance() {
        return balance;
    }

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

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

}
```

```
package com.icin.entity;
```

```java
import java.time.LocalDate;

import jakarta.persistence.Entity;
import  jakarta.persistence.GeneratedValue;
import  jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table
public class ChequebookRequest {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private long account;
    private String accType;
    private boolean requestStatus;
    private LocalDate date;
    private int no_of_pages;

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


    public long getAccount() {
        return account;
    }
    public void setAccount(long account) {
        this.account = account;
    }
    public String getAccType() {
        return accType;
    }
    public void setAccType(String accType) {
        this.accType = accType;
    }
    public boolean isRequestStatus() {
        return requestStatus;
    }
    public void setRequestStatus(boolean requestStatus) {
        this.requestStatus = requestStatus;
    }
    public LocalDate getDate() {
        return date;
    }
    public void setDate(LocalDate date) {
        this.date = date;
    }
    public int getNo_of_pages() {
        return no_of_pages;
    }
    public void setNo_of_pages(int no_of_pages) {
        this.no_of_pages = no_of_pages;
    }
```

```
}
```

```
package com.icin.entity;

import   jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinColumn;
import jakarta.persistence.OneToOne;
import jakarta.persistence.Table;


@Entity
@Table
public class Saccount {

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

    private long accno;

    private int balance;

    private String username;

    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name="user_id")
    private User user;

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }

    public String getUsername() {
        return username;
    }

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

    public int getId() {
        return id;
    }

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

    public long getAccno() {
```

```
        return accno;
    }

    public void setAccno(long accno) {
        this.accno = accno;
    }

    public int getBalance() {
        return balance;
    }

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


}
```

```
package com.icin.entity;

import java.time.LocalDate;

import jakarta.persistence.Entity;
import  jakarta.persistence.GeneratedValue;
import  jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;

@Entity
@Table
public class Transfer implements Comparable<Transfer>{

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private long saccount;
    private long raccount;
    private int amount;
    private LocalDate date;


    @Override
    public int compareTo(Transfer o) {
        // TODO Auto-generated method stub
        Integer i1=this.id;
        Integer i2=o.id;
        return i2.compareTo(i1);    }


    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public long getSaccount() {
        return saccount;
```

```
    }
    public void setSaccount(long saccount) {
        this.saccount = saccount;
    }
    public long getRaccount() {
        return raccount;
    }
    public void setRaccount(long raccount) {
        this.raccount = raccount;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
    public LocalDate getDate() {
        return date;
    }
    public void setDate(LocalDate date) {
        this.date = date;
    }

}
```

```
package com.icin.entity;

import java.sql.Date;

import com.fasterxml.jackson.annotation.JsonFormat;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.FetchType;
import  jakarta.persistence.GeneratedValue;
import  jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import  jakarta.persistence.OneToOne;
import jakarta.persistence.Table;

@Entity
@Table(name = "user")
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private  int  id;
    private String fname;
    private String lname;
    private long phone;
    private String address;
    private String email;
    private String username;
    private String password;

    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern = "dd-MM-yyyy")
```

```
    private Date dob;

    private String identityType;

    private String identity;

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

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

    @Column(columnDefinition = "integer default 3", nullable = false)
    private int featureStatus = 3;

    @OneToOne(targetEntity = Account.class, mappedBy = "user", orphanRemoval = false, fetch =
FetchType.LAZY)
    private Account account;

    @OneToOne(targetEntity = Saccount.class, mappedBy = "user", orphanRemoval = false, fetch =
FetchType.LAZY)
    private Saccount sAccount;

    public User(int id, String fname, String lname, long phone, String address, String email, String
username,
            String password, Date dob, String identityType, String identity, boolean status,
            boolean authorizationStatus, int featureStatus, Account account, Saccount sAccount) {
        super();
        this.id = id;
        this.fname = fname;
        this.lname = lname;
        this.phone = phone;
        this.address = address;
        this.email = email;
        this.username = username;
        this.password = password;
        this.dob = dob;
        this.identityType = identityType;
        this.identity = identity;
        this.status = status;
        this.authorizationStatus = authorizationStatus;
        this.featureStatus = featureStatus;
        this.account = account;
        this.sAccount = sAccount;
    }

    public int getId() {
        return id;
    }

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

    public String getFname() {
        return fname;
    }

    public void setFname(String fname) {
```

```
        this.fname = fname;
    }

    public String getLname() {
        return lname;
    }

    public void setLname(String lname) {
        this.lname = lname;
    }

    public long getPhone() {
        return phone;
    }

    public void setPhone(long phone) {
        this.phone = phone;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    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 Date getDob() {
        return dob;
    }

    public void setDob(Date dob) {
        this.dob = dob;
    }
```

```
public String getIdentityType() {
    return identityType;
}

public void setIdentityType(String identityType) {
    this.identityType = identityType;
}

public String getIdentity() {
    return identity;
}

public void setIdentity(String identity) {
    this.identity = identity;
}

public boolean getStatus() {
    return status;
}

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

public boolean isAuthorizationStatus() {
    return authorizationStatus;
}

public void setAuthorizationStatus(boolean authorizationStatus) {
    this.authorizationStatus = authorizationStatus;
}

public int getFeatureStatus() {
    return featureStatus;
}

public void setFeatureStatus(int featureStatus) {
    this.featureStatus = featureStatus;
}

public Account getAccount() {
    return account;
}

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

public Saccount getsAccount() {
    return sAccount;
}

public void setsAccount(Saccount sAccount) {
    this.sAccount = sAccount;
}

public User() {
    super();
    // TODO Auto-generated constructor stub
```

```
    }

}
```

```
package com.icin.entity;

public class UserDisplay {

    private String fname;
    private String lname;
    private long phone;
    private String username;
    private boolean status;
    private int featureStatus;
    private long primaryAccno;
    private int primaryBalance;
    private long savingsAccno;
    private int savingsBalance;



    public UserDisplay(String fname, String lname, long phone, String username, boolean status, int
featureStatus,
            long primaryAccno, int primaryBalance, long savingsAccno, int savingsBalance) {
        super();
        this.fname = fname;
        this.lname = lname;
        this.phone = phone;
        this.username = username;
        this.status = status;
        this.featureStatus = featureStatus;
        this.primaryAccno = primaryAccno;
        this.primaryBalance = primaryBalance;
        this.savingsAccno = savingsAccno;
        this.savingsBalance = savingsBalance;
    }



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



    public String getFname() {
        return fname;
    }



    public void setFname(String fname) {
        this.fname = fname;
    }
```

```java
public String getLname() {
    return lname;
}



public void setLname(String lname) {
    this.lname = lname;
}



public long getPhone() {
    return phone;
}



public void setPhone(long phone) {
    this.phone = phone;
}



public String getUsername() {
    return username;
}



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



public boolean isStatus() {
    return status;
}



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



public int getFeatureStatus() {
    return featureStatus;
}



public void setFeatureStatus(int featureStatus) {
    this.featureStatus = featureStatus;
}
```

```
    public long getPrimaryAccno() {
        return primaryAccno;
    }



    public void setPrimaryAccno(long primaryAccno) {
        this.primaryAccno = primaryAccno;
    }



    public int getPrimaryBalance() {
        return primaryBalance;
    }



    public void setPrimaryBalance(int primaryBalance) {
        this.primaryBalance = primaryBalance;
    }



    public long getSavingsAccno() {
        return savingsAccno;
    }



    public void setSavingsAccno(long savingsAccno) {
        this.savingsAccno = savingsAccno;
    }



    public int getSavingsBalance() {
        return savingsBalance;
    }



    public void setSavingsBalance(int savingsBalance) {
        this.savingsBalance = savingsBalance;
    }

}
```

```
package com.icin.entity;

import java.time.LocalDate;

import jakarta.persistence.Entity;
import  jakarta.persistence.GeneratedValue;
import  jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
```

```java
import jakarta.persistence.Table;

@Entity
@Table
public class UserHistory {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private int id;
    private long account;
    private int amount;
    private String action;
    private LocalDate date;
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public long getAccount() {
        return account;
    }
    public void setAccount(long account) {
        this.account = account;
    }
    public int getAmount() {
        return amount;
    }
    public void setAmount(int amount) {
        this.amount = amount;
    }
    public String getAction() {
        return action;
    }
    public void setAction(String action) {
        this.action = action;
    }
    public LocalDate getDate() {
        return date;
    }
    public void setDate(LocalDate date) {
        this.date = date;
    }

}
```

```java
package com.icin.repository;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.Account;

@Repository
public interface AccountRepository  extends CrudRepository<Account, Integer>{

    public Account findByUsername(String username);
```

```
    public Account findByAccno(long accno);

}
```

```
package com.icin.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.ChequebookRequest;

@Repository
public interface ChequeBookRepository extends JpaRepository<ChequebookRequest, Integer> {

    List<ChequebookRequest> findByAccount(long account);

}
```

```
package com.icin.repository;

import java.util.List;

import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.ChequebookRequest;

import jakarta.transaction.Transactional;

@Repository
public interface ChequeBookRequestsRepository extends CrudRepository<ChequebookRequest, Integer>{

    @Modifying
    @Transactional
// @Query("update ChequebookRequest c set c.requestStatus=1 where c.account = ?1")
    @Query("update ChequebookRequest c set c.requestStatus=true where c.account = ?1")
    void setChequebookInfoByAccount(long accNo);

    @Query("FROM ChequebookRequest c where c.requestStatus=FALSE")
    public List<ChequebookRequest> findAllChequebookRequests();


    public List<ChequebookRequest> findByAccount(long account);

}
```

```
package com.icin.repository;

import org.springframework.data.repository.CrudRepository;
```

```java
import org.springframework.stereotype.Repository;

import com.icin.entity.Saccount;

@Repository
public interface SaccountRepository extends CrudRepository<Saccount, Integer>{

    public Saccount findByAccno(long accNo);

    public Saccount findByUsername(String username);

}
```

```java
package com.icin.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.Transfer;

@Repository
public interface TransferHistoryRepository extends JpaRepository<Transfer, Integer> {

    List<Transfer> findBySaccount(long saccount);

    List<Transfer> findByRaccount(long raccount);

}
```

```java
package com.icin.repository;

import java.util.List;

import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.Transfer;

@Repository
public interface TransferRepository extends CrudRepository<Transfer, Integer>{

    public List<Transfer> findBySaccount(long saccount);
    public List<Transfer> findByRaccount(long racoount);

}
```

```java
package com.icin.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
```

```
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import com.icin.entity.User;
import com.icin.entity.UserDisplay;

@Repository
public interface UserDisplayRepository extends JpaRepository<User, Integer>{

    @Query("SELECT new
com.icin.entity.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance
,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE u.username=a.username and
u.username=s.username")
    public List<UserDisplay> getAllUsers();

    @Query("SELECT new
com.icin.entity.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance
,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE u.username=?1 and u.username=a.username
and u.username=s.username")
    public UserDisplay getUserDetailsByUsername(String userDetail);

    @Query("SELECT new
com.icin.entity.UserDisplay(u.fname,u.lname,u.phone,u.username,u.status,u.featureStatus,a.accno,a.balance
,s.accno,s.balance)" + "FROM User u ,Account a,Saccount s WHERE s.accno=?1 and u.username=a.username and
u.username=s.username")
    public UserDisplay getUserDetailsByAccountNo(long accNo);


    @Query("SELECT new com.icin.entity.UserDisplay(u.username,a.accno,a.balance,s.accno,s.balance)" +
"FROM User u ,Account a,Saccount s WHERE u.username=?1 and u.username=a.username and
u.username=s.username")
    public UserDisplay getCurrentUser(String username);

}
```

```
package com.icin.repository;

import java.util.List;

import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.UserHistory;

@Repository
public interface UserHistoryRepository extends JpaRepository<UserHistory, Integer>{

    public List<UserHistory> findByAccount(long account);

}
```

```
package com.icin.repository;

import java.util.List;
```

```java
import org.springframework.data.jpa.repository.Modifying;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;

import com.icin.entity.User;

import jakarta.transaction.Transactional;

@Repository
public interface UserRepository extends CrudRepository<User, Integer> {

    @Query("FROM User u WHERE u.username=?1")
    public User findByUsername(String username);

    @Modifying
    @Transactional
    @Query("update User u set u.status=true where u.username = ?1")
    void disableUser(String username);

    @Modifying
    @Transactional
    @Query("update User u set u.status=false where u.username = ?1")
    void enableUser(String username);

    @Modifying
    @Transactional
    @Query("update User u set u.authorizationStatus=true where u.username = ?1")
    void authorizeUser(String username);

    @Modifying
    @Transactional
    @Query("delete from User u where u.username = ?1")
    void cancelAuthorization(String username);

    @Query("FROM User u where u.authorizationStatus=FALSE")
    public List<User> findAllUnauthorizedAccounts();

    @Modifying
    @Transactional
    @Query("update User u set u.featureStatus=?2 where u.username = ?1")
    void setUserFeatureStatus(String username, int featureId);


    public User findByEmail(String email);


    public User findByPhone(long l);

}
```

```java
package com.icin.response;

public class ChequeResponse {

    private boolean status;
       private String responseMessage;
       private long account;
```

```
    public ChequeResponse() {
    }

    public boolean isStatus() {
        return this.status;
    }

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

    public String getResponseMessage() {
        return this.responseMessage;
    }

    public void setResponseMessage(String responseMessage) {
        this.responseMessage = responseMessage;
    }

    public long getAccount() {
        return this.account;
    }

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

}
```

```
package com.icin.response;

public class DepositResponse {

    private boolean depositStatus;
    private String responseMessage;
    private long account;

    public DepositResponse() {
    }

    public boolean isDepositStatus() {
        return this.depositStatus;
    }

    public void setDepositStatus(boolean depositStatus) {
        this.depositStatus = depositStatus;
    }

    public String getResponseMessage() {
        return this.responseMessage;
    }

    public void setResponseMessage(String responseMessage) {
        this.responseMessage = responseMessage;
    }
```

```
        public long getAccount() {
            return this.account;
        }

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

}
```

```
package com.icin.response;

public class LoginResponse {

    private boolean loginStatus;
        private String responseMessage;
        private String username;

        public LoginResponse() {
        }

        public boolean getLoginStatus() {
            return this.loginStatus;
        }

        public void setLoginStatus(boolean loginStatus) {
            this.loginStatus = loginStatus;
        }

        public String getResponseMessage() {
            return this.responseMessage;
        }

        public void setResponseMessage(String responseMessage) {
            this.responseMessage = responseMessage;
        }

        public String getUsername() {
            return this.username;
        }

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

}
```

```
package     com.icin.response;

public class RegisterResponse {

    private boolean registrationStatus;
        private String responseMessage;
        private String username;
```

```
        public RegisterResponse() {
        }

        public boolean getRegistrationStatus() {
            return this.registrationStatus;
        }

        public void setRegistrationStatus(boolean registrationStatus) {
            this.registrationStatus = registrationStatus;
        }

        public String getResponseMessage() {
            return this.responseMessage;
        }

        public void setResponseMessage(String responseMessage) {
            this.responseMessage = responseMessage;
        }

        public String getUsername() {
            return this.username;
        }

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

}
```

```
package com.icin.response;

public class TransferResponse {

private boolean transferStatus; private String responseMessage; private long
saccount;

public TransferResponse() {
}

public boolean isTransferStatus() { return this.transferStatus;
}

public void setTransferStatus(boolean transferStatus) { this.transferStatus =
transferStatus;
}

public String getResponseMessage() { return this.responseMessage;
}

public void setResponseMessage(String responseMessage) { this.responseMessage =
responseMessage;
}

public long getSaccount() {
```

```
        return this.saccount;
    }

    public void setSaccount(long saccount) {
        this.saccount = saccount;
    }

}
```

```
package com.icin.response;

public class UpdateResponse {

    public boolean flag;
    public String message;

    public UpdateResponse() {
    }

    public boolean isFlag() {
        return this.flag;
    }

    public void setFlag(boolean flag) {
        this.flag = flag;
    }

    public String getMessage() {
        return this.message;
    }

    public void setMessage(String message) {
        this.message = message;
    }

}
```

```
package com.icin.response;

public class WithdrawResponse {

private boolean withdrawStatus; private String responseMessage; private long
account;

public WithdrawResponse() {
}

public boolean isWithdrawStatus() { return this.withdrawStatus;
}

public void setWithdrawStatus(boolean withdrawStatus) { this.withdrawStatus =
withdrawStatus;
}
```

```
    public String getResponseMessage() {
        return this.responseMessage;
    }

    public void setResponseMessage(String responseMessage) {
        this.responseMessage = responseMessage;
    }

    public long getAccount() {
        return this.account;
    }

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

}
```

```
package com.icin.service;

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

import com.icin.entity.Account;
import com.icin.entity.Saccount;
import com.icin.entity.User;
import com.icin.repository.AccountRepository;
import com.icin.repository.SaccountRepository;
import com.icin.repository.UserRepository;
import com.icin.response.*;

@Service
public class AccountsService {

    @Autowired
    private AccountRepository accountRepo;

    @Autowired
    private UserHistoryService service;

    @Autowired
    private TransferHistoryService tservice;

    @Autowired
    private UserRepository userRepo;

    @Autowired
    private SaccountRepository sdao;

    private final String bankCode = "3914";
    private final String countryCode = "91";
    private final String branchCode = "820";
    private final String accountcode = "1";

    public long generate_saving(int userId) {
        String accNo = "3914918201" + String.valueOf(userId);
```

```
        return Long.parseLong(accNo);
    }

    public static boolean isprimary(long account) {
        String s = Long.toString(account).substring(0, 10);
        String check = "3914918201";
        return s.equals(check);
    }

    public Account newAccount(String username, int userId) {
        Account account = new Account();
        account.setUsername(username);
        account.setAccno(this.generate_saving(userId));
        account.setUser(this.userRepo.findByUsername(username));
        return (Account) this.accountRepo.save(account);
    }

    public Account getAccount(String username) {
        return this.accountRepo.findByUsername(username);
    }

    public DepositResponse deposit(long acc, int amount) {
        DepositResponse response = new DepositResponse();
        boolean flag = true;

        try {
            Account account = this.accountRepo.findByAccno(acc);
            account.setBalance(account.getBalance() + amount);
            this.service.addAction(acc, amount, account.getBalance(), "credit");
            this.accountRepo.save(account);
            response.setResponseMessage("Rs." + amount + " successfully deposited into your account
balance is now Rs."
                    + account.getBalance());
            response.setDepositStatus(flag);
        } catch (Exception var7) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
            response.setDepositStatus(flag);
        }

        response.setAccount(acc);
        return response;
    }

    public WithdrawResponse withdraw(long acc, int amount) {
        WithdrawResponse response = new WithdrawResponse();
        boolean flag = true;

        try {
            Account account = this.accountRepo.findByAccno(acc);
            User user = this.userRepo.findByUsername(account.getUsername());
            if (user.getFeatureStatus() != 2 && user.getFeatureStatus() != 3) {
                flag = false;
                response.setResponseMessage("This function is not available for your account");
                response.setWithdrawStatus(flag);
            } else if (account.getBalance() >= amount) {
                account.setBalance(account.getBalance() - amount);
                this.service.addAction(acc, amount, account.getBalance(), "debit");
                this.accountRepo.save(account);
```

```
                response.setResponseMessage("Rs." + amount + " successfully withdrawn your account
balance is now Rs."
                        + account.getBalance());
                response.setWithdrawStatus(flag);
            } else {
                flag = false;
                response.setResponseMessage("Insufficient funds to complete the transaction");
                response.setWithdrawStatus(flag);
            }
        } catch (Exception var8) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
            response.setWithdrawStatus(flag);
        }

        response.setAccount(acc);
        return response;
    }

    public TransferResponse transfer(long saccount, long raccount, int amount) {
        TransferResponse response = new TransferResponse();
        boolean flag = true;

        try {
            Account senderAccount = this.accountRepo.findByAccno(saccount);
            User user;
            if (isprimary(raccount)) {
                Account receiverAccount = this.accountRepo.findByAccno(raccount);
                if (senderAccount.getAccno() != receiverAccount.getAccno()) {
                    if (senderAccount.getBalance() > amount) {
                        user = this.userRepo.findByUsername(senderAccount.getUsername());
                        if (user.getFeatureStatus() == 3) {
                            senderAccount.setBalance(senderAccount.getBalance() - amount);
                            receiverAccount.setBalance(receiverAccount.getBalance() + amount);
                            this.tservice.addAction(saccount, raccount, amount);
                            this.accountRepo.save(senderAccount);
                            this.accountRepo.save(receiverAccount);
                            response.setResponseMessage("Rs." + amount + " successfully transferred to
account "
                                    + receiverAccount.getAccno());
                            response.setTransferStatus(flag);
                        } else {
                            flag = false;
                            response.setResponseMessage("This feature is not available for your
account");
                            response.setTransferStatus(flag);
                        }
                    } else {
                        flag = false;
                        response.setResponseMessage("Insufficient funds to complete the transfer");
                        response.setTransferStatus(flag);
                    }
                } else {
                    flag = false;
                    response.setResponseMessage("sender and recieiver accounts are same");
                    response.setTransferStatus(flag);
                }
            } else {
                Saccount receiverAccount = this.sdao.findByAccno(raccount);
```

```
                if (senderAccount.getAccno() != receiverAccount.getAccno()) {
                    if (senderAccount.getBalance() > amount) {
                        user = this.userRepo.findByUsername(senderAccount.getUsername());
                        if (user.getFeatureStatus() == 3) {
                            senderAccount.setBalance(senderAccount.getBalance() - amount);
                            receiverAccount.setBalance(receiverAccount.getBalance() + amount);
                            this.tservice.addAction(saccount, raccount, amount);
account "                   this.accountRepo.save(senderAccount);
                            this.sdao.save(receiverAccount);
                            response.setResponseMessage("Rs." + amount + " successfully transferred to

                                    + receiverAccount.getAccno());
                            response.setTransferStatus(flag);
                        } else {
                            flag = false;
                            response.setResponseMessage("This function isnt available for the account");
                            response.setTransferStatus(flag);
                        }
                    } else {
                        flag = false;
                        response.setResponseMessage("Insufficient funds to complete the transfer");
                        response.setTransferStatus(flag);
                    }
                } else {
                    flag = false;
                    response.setResponseMessage("sender and recieiver accounts are same");
                    response.setTransferStatus(flag);
                }
            }
        } catch (Exception var11) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
            response.setTransferStatus(flag);
        }

        response.setSaccount(saccount);
        return response;
    }

    public Account getAccountDetails(long account) {
        return this.accountRepo.findByAccno(account);
    }

    public Account updateAccount(Account account) {
        return (Account) this.accountRepo.save(account);
    }

}
```

```
package com.icin.service;

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

import org.apache.commons.mail.EmailException;
import org.springframework.beans.factory.annotation.Autowired;
```

```java
import org.springframework.stereotype.Service;

import com.icin.entity.ChequebookRequest;
import com.icin.entity.Transfer;
import com.icin.entity.User;
import com.icin.entity.UserDisplay;
import com.icin.repository.AccountRepository;
import com.icin.repository.ChequeBookRequestsRepository;
import com.icin.repository.SaccountRepository;
import com.icin.repository.TransferRepository;
import com.icin.repository.UserDisplayRepository;
import com.icin.repository.UserRepository;

@Service
public class AdminService {

    @Autowired
    private TransferRepository transferRepository;

    @Autowired
    private ChequeBookRequestsRepository chequeBookRequestsRepository;

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private UserDisplayRepository userDisplayRepository;

    @Autowired
    private AccountRepository accountRepository;

    @Autowired
    private SaccountRepository aSaccountRepository;

    @Autowired
    private MailService mailService;

    @Autowired
    private AccountsService accService;

    @Autowired
    private SaccountCreationService sAccService;

    public void authorizeUser(String username) {
        userRepository.authorizeUser(username);
        System.out.println("error here top");
        User currUser = userRepository.findByUsername(username);
        int userId = currUser.getId();
        System.out.println("error here 1");
        accService.newAccount(username, userId);
        System.out.println("error here 2");
        sAccService.newAccount(username, userId);
    }

    public void cancelAuthorization(String username) {
        userRepository.cancelAuthorization(username);
    }

    public void acceptChequebookRequest(long accNo) {
```

```
        String username = "";
        chequeBookRequestsRepository.setChequebookInfoByAccount(accNo);
        if (Long.toString(accNo).length() == 7) {
            username = accountRepository.findByAccno(accNo).getUsername();
        } else {
            username = aSaccountRepository.findByAccno(accNo).getUsername();
        }
        try {
            mailService.sendChequebookConfirmedEmail(username);
        } catch (EmailException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }

    public void enableUser(String username) {
        userRepository.enableUser(username);

    }

    public void disableUser(String username) {
        userRepository.disableUser(username);

    }

    public List<UserDisplay> getAllUsers() {
        return userDisplayRepository.getAllUsers();
    }

    public List<Transfer> getAllTransactions(long accountNo) {
        List<Transfer> sender = transferRepository.findBySaccount(accountNo);
        List<Transfer> receiver = transferRepository.findByRaccount(accountNo);
        List<Transfer> merged = new ArrayList<>();
        merged.addAll(sender);
        merged.addAll(receiver);
        Collections.sort(merged);
        return merged;
    }

    public List<ChequebookRequest> getAllChequebookRequests() {
        return chequeBookRequestsRepository.findAllChequebookRequests();
    }


    public List<User> getAllUnauthorizedUsers() {
        return userRepository.findAllUnauthorizedAccounts();
    }


    public void setUserFeatures(String username, int featureId) {
        userRepository.setUserFeatureStatus(username,  featureId);

    }

    static boolean isNumber(String s) {
        for (int i = 0; i < s.length(); i++)
            if (Character.isDigit(s.charAt(i)) == false)
                return false;
        return true;
```

```
    }


    public UserDisplay searchUser(String userDetail) {
        if (isNumber(userDetail)) {
            return userDisplayRepository.getUserDetailsByAccountNo(Long.parseLong(userDetail));
        } else {
            return userDisplayRepository.getUserDetailsByUsername(userDetail);
        }

    }

}
```

```
package com.icin.service;

import java.time.LocalDate;
import java.util.List;

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

import com.icin.entity.Account;
import com.icin.entity.ChequebookRequest;
import com.icin.entity.Saccount;
import com.icin.repository.AccountRepository;
import com.icin.repository.ChequeBookRepository;
import com.icin.repository.SaccountRepository;
import com.icin.response.ChequeResponse;

@Service
public class ChequebookService {

    @Autowired
    private ChequeBookRepository dao;

    @Autowired
    private AccountRepository aRepository ;

    @Autowired
    private SaccountRepository sRepository;


    public ChequeResponse createrequest(ChequebookRequest chequebook) {
        ChequeResponse response = new ChequeResponse();
        long account = chequebook.getAccount();
        List<ChequebookRequest> prevRequests = this.dao.findByAccount(account);
        if (!prevRequests.isEmpty()) {
            for (int i = 0; i < prevRequests.size(); ++i) {
                if (!((ChequebookRequest) prevRequests.get(i)).isRequestStatus()) {
                    response.setResponseMessage("Your previous chequebook request is still pending.");
                    response.setStatus(false);
                    response.setAccount(account);
                    return response;
                }
            }
        }
```

```
        LocalDate today = LocalDate.now();
        if (isprimary(account)) {
            try {
                Account account1 = this.aRepository.findByAccno(account);
                response.setAccount(account1.getAccno());
                response.setStatus(true);
                response.setResponseMessage("Request submitted successfully");
                chequebook.setAccType("Primary");
                chequebook.setDate(today);
                chequebook.setRequestStatus(false);
                this.dao.save(chequebook);
            } catch (Exception var9) {
                response.setAccount(account);
                response.setStatus(false);
                response.setResponseMessage("account number is incorrect");
            }
        } else if (isSecondary(account)) {
            try {
                Saccount saccount = this.sRepository.findByAccno(account);
                response.setAccount(saccount.getAccno());
                response.setStatus(true);
                response.setResponseMessage("Request submitted successfully");
                chequebook.setRequestStatus(false);
                chequebook.setAccType("Secondary");
                chequebook.setDate(today);
                this.dao.save(chequebook);
            } catch (Exception var8) {
                response.setAccount(account);
                response.setStatus(false);
                response.setResponseMessage("account number is incorrect");
            }
        } else {
            response.setAccount(account);
            response.setStatus(false);
            response.setResponseMessage("account number is incorrect");
        }

        return response;
    }

    public List<ChequebookRequest> getRequests(long account) {
        return this.dao.findByAccount(account);
    }

    public static boolean isprimary(long account) {
        String s = Long.toString(account).substring(0, 10);
        String check = "3914918201";
        return s.equals(check);
    }

    public static boolean isSecondary(long account) {
        String s = Long.toString(account).substring(0, 10);
        String check = "3914918202";
        return s.equals(check);
    }

}
```

```java
package com.icin.service;

import org.apache.commons.codec.digest.DigestUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.icin.details.LoginDetails;
import com.icin.entity.User;
import com.icin.repository.UserRepository;
import com.icin.response.LoginResponse;

@Service
public class LoginService {

    @Autowired
    private UserRepository uRepository;

    public LoginResponse customerLogin(LoginDetails login) {
        LoginResponse response = new LoginResponse();
        boolean flag = true;
        String message = "Login succesfull";
        User user = null;
        String hashedPassword = DigestUtils.sha256Hex(login.getPassword());

        try {
            user = this.uRepository.findByUsername(login.getUsername());
            if (user.getStatus()) {
                flag = false;
                message = "Dear Mr/Ms." + user.getFname() + " your account has been blocked for security
reasons.";
            }

            if (!user.isAuthorizationStatus()) {
                flag = false;
                message = "Dear Mr/Ms." + user.getFname() + " your account has not been activated yet";
            }

            if (!hashedPassword.equals(user.getPassword())) {
                flag = false;
                message = "Username or password is incorrect";
            }
        } catch (Exception var9) {
            flag = false;
            message = "Username or password is incorrect";
        }

        response.setLoginStatus(flag);
        response.setResponseMessage(message);

        try {
            response.setUsername(user.getUsername());
        } catch (Exception var8) {
            response.setUsername(login.getUsername());
        }
        return response;
    }
}
```

```java
package com.icin.service;

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

import com.icin.entity.Account;
import com.icin.entity.User;
import com.icin.repository.AccountRepository;
import com.icin.repository.UserRepository;

import org.apache.commons.mail.DefaultAuthenticator;
import org.apache.commons.mail.EmailException;
import org.apache.commons.mail.HtmlEmail;

@Service
public class MailService {

    @Autowired
    private UserRepository userRepository;

    @Autowired
    private AccountRepository accountRepository;

    public void sendAuthorizedEmail(String username) throws EmailException {

        User user = userRepository.findByUsername(username);
        String receiver = user.getEmail();
        String fullName = user.getFname() + " " + user.getLname();
        Account account = accountRepository.findByUsername(username);
        long accountNo = account.getAccno();
        int balance = account.getBalance();

        HtmlEmail mail = new HtmlEmail();
        // mail.setSSL(true);
        mail.setHostName("smtp.gmail.com");
        mail.setSmtpPort(465);
        mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));

        mail.setFrom("icinbank.noreply@gmail.com", "ICIN");
        mail.addTo(receiver, "To");
        mail.setSubject(user.getFname() + ", Welcome to your new ICIN Bank Account");
        mail.setHtmlMsg("<h1 style='font-size:27px;color:blue'>Hi " + fullName + ",</h1>"
                + "<br><h2 style='font-size:21px;color:black'>Thank you for creating a ICIN Bank account.
Your account has been activated,"
                + " Here are your details.</h2><br>" + "<p style='font-size:18px;'>Username: " + username
                + "<br><p style='font-size:18px;'>Account Number: " + accountNo + "<br>"
                + "<p style='font-size:18px;'>Current Balance: Rs. " + balance + "</p>");
        mail.send();
        System.out.println("email sent");
    }

    public void sendChequebookConfirmedEmail(String username) throws EmailException {

        User user = userRepository.findByUsername(username);
        String receiver = user.getEmail();
        String fullName = user.getFname() + " " + user.getLname();
```

```
        HtmlEmail mail = new HtmlEmail();
        // mail.setSSL(true);
        mail.setHostName("smtp.gmail.com");
        mail.setSmtpPort(465);
        mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));
        mail.setFrom("icinbank.noreply@gmail.com", "ICIN");
        mail.addTo(receiver, "To");
        mail.setSubject("ICIN Bank Chequebook Request Confirmed");
        mail.setHtmlMsg("<h1 style='font-size:25px;color:green'>Hi " + fullName + ",</h1>"
                + "<br><p style='font-size:19px;color:black'>Your chequebook request has been processed
and confirmed it will be sent to you via courier in 4-5 business days.</p><br><br>");
        mail.send();
        System.out.println("email sent");
    }

    public void sendAuthorizeCancelEmail(String username) throws EmailException {

        User user = userRepository.findByUsername(username);
        String receiver = user.getEmail();
        String fullName = user.getFname() + " " + user.getLname();

        HtmlEmail mail = new HtmlEmail();
        // mail.setSSL(true);
        mail.setHostName("smtp.gmail.com");
        mail.setSmtpPort(465);
        mail.setAuthenticator(new DefaultAuthenticator("icinbank.noreply@gmail.com", "icin@1155"));

        mail.setFrom("icinbank.noreply@gmail.com", "ICIN");
        mail.addTo(receiver, "To");
        mail.setSubject("ICIN Bank Account Cancelled");
        mail.setHtmlMsg("<h1 style='font-size:27px;color:red'>Hi " + fullName + ",</h1>"
                + "<br><p style='font-size:19px;color:black'>Sorry, Your account with ICIN Bank with
username "
                 + username + " has been cancelled. Please contact nearest branch for more
details.</p><br><br>");
        mail.send();
        System.out.println("email sent");
    }

}
```

```
package com.icin.service;

import org.apache.commons.codec.digest.DigestUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.icin.details.UpdateDetails;
import com.icin.entity.User;
import com.icin.entity.UserDisplay;
import com.icin.repository.UserDisplayRepository;
import com.icin.repository.UserRepository;
import com.icin.response.UpdateResponse;

@Service
public class ProfileService {
```

```
    @Autowired
    private UserRepository userRepository;

    @Autowired
    private UserDisplayRepository userDisplayRepository;

    public UpdateResponse updateUser(UpdateDetails user) {
        boolean flag = true;
        UpdateResponse response = new UpdateResponse();
        String message = "Update successful";

        try {
            int counter = 0;
            User u = this.userRepository.findByUsername(user.getUsername());
            if (user.getAddress().length() != 0) {
                ++counter;
                u.setAddress(user.getAddress());
            }

            if (user.getPassword().length() != 0 && user.getNewpassword().length() != 0) {
                ++counter;
                String hashedPassword = DigestUtils.sha256Hex(user.getPassword());
                u.setPassword(hashedPassword);
            }

            if (user.getEmail().length() != 0) {
                ++counter;
                u.setEmail(user.getEmail());
            }

            if (user.getPhone() != 0L) {
                ++counter;
                u.setPhone(user.getPhone());
            }

            System.out.println(counter);
            if (counter > 0) {
                this.userRepository.save(u);
            } else {
                flag = false;
                message = "Please enter some information to update";
            }
        } catch (Exception var8) {
            flag = false;
            response.setMessage("Update unsuccesful");
        }

        response.setMessage(message);
        response.setFlag(flag);
        return response;
    }

    public User getUser(String username) {
        return this.userRepository.findByUsername(username);
    }

    public UserDisplay userDisplay(String username) {
        UserDisplay user = this.userDisplayRepository.getCurrentUser(username);
```

```
        return user;
    }

}
```

```
package com.icin.service;

import org.apache.commons.codec.digest.DigestUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.icin.entity.User;
import com.icin.repository.UserRepository;
import com.icin.response.RegisterResponse;

@Service
public class RegistrationService {

    @Autowired
    private UserRepository dao;

    public RegisterResponse createAccount(User user) {
        RegisterResponse response = new RegisterResponse();
        boolean flag = true;
        String message = "Registration Succesful";
        if (this.EmailAlreadyExists(user.getEmail())) {
            message = "Email already Exists";
            flag = false;
        }

        if (this.PhoneAlreadyExists(user.getPhone())) {
            message = "Phone number already Exists";
            flag = false;
        }

        if (this.usernameAlreadyExists(user.getUsername())) {
            message = "Username already Exists";
            flag = false;
        }

        if (flag) {
            String hashedPassword = DigestUtils.sha256Hex(user.getPassword());
            user.setPassword(hashedPassword);
            this.dao.save(user);
        }

        response.setRegistrationStatus(flag);
        response.setResponseMessage(message);
        response.setUsername(user.getUsername());
        return response;
    }

    public boolean usernameAlreadyExists(String username) {
        try {
            User u = this.dao.findByUsername(username);
            System.out.println(u.toString());
            return true;
```

```
        } catch (Exception var3) {
            return false;
        }
    }

    public boolean EmailAlreadyExists(String email) {
        try {
            User u = this.dao.findByEmail(email);
            System.out.println(u.toString());
            return true;
        } catch (Exception var3) {
            return false;
        }
    }

    public boolean PhoneAlreadyExists(long l) {
        try {
            User u = this.dao.findByPhone(l);
            System.out.println(u.toString());
            return true;
        } catch (Exception var4) {
            return false;
        }
    }

}
```

```
package com.icin.service;

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

import com.icin.entity.Saccount;
import com.icin.repository.SaccountRepository;
import com.icin.repository.UserRepository;

@Service
public class SaccountCreationService {

    @Autowired
    private SaccountRepository saccountRepository;

    @Autowired
    private UserRepository userRepository;

    private final String bankCode = "3914";
    private final String countryCode = "91";
    private final String branchCode = "820";
    private final String accountcode = "2";

    public long generate_saving(int userId) {
        String accNo = bankCode + countryCode + branchCode + accountcode + String.valueOf(userId);
        return Long.parseLong(accNo);
    }

    public Saccount newAccount(String username, int userId) {
        Saccount account = new Saccount();
```

```
        account.setUsername(username);
        account.setAccno(generate_saving(userId));
        account.setUser(userRepository.findByUsername(username));
        return saccountRepository.save(account);
    }

}
```

```
package com.icin.service;

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

import com.icin.entity.Account;
import com.icin.entity.Saccount;
import com.icin.entity.User;
import com.icin.repository.AccountRepository;
import com.icin.repository.SaccountRepository;
import com.icin.repository.UserRepository;
import com.icin.response.*;
//import com.icin.response.WithdrawResponse;

@Service
public class SavingsAccountService {

    @Autowired
    private SaccountRepository saccountRepository;

    @Autowired
    private UserHistoryService service;

    @Autowired
    private TransferHistoryService tservice;

    @Autowired
    private AccountRepository adao;

    @Autowired
    private UserRepository udao;

    private final String bankCode = "3914";
    private final String countryCode = "91";
    private final String branchCode = "820";
    private final String accountcode = "2";

    public Saccount getAccount(String username) {
        return this.saccountRepository.findByUsername(username);
    }

    public Saccount getAccountDetails(long account) {
        return this.saccountRepository.findByAccno(account);
    }

    public long generate_saving(int userId) {
        String accNo = "3914918202" + String.valueOf(userId);
        return Long.parseLong(accNo);
    }
```

```
    public Saccount newAccount(String username, int userId) {
        Saccount account = new Saccount();
        account.setUsername(username);
        account.setAccno(this.generate_saving(userId));
        account.setUser(this.udao.findByUsername(username));
        return (Saccount) this.saccountRepository.save(account);
    }

    public DepositResponse deposit(long acc, int amount) {
        DepositResponse response = new DepositResponse();
        boolean flag = true;

        try {
            Saccount account = this.saccountRepository.findByAccno(acc);
            account.setBalance(account.getBalance() + amount);
            this.service.addAction(acc, amount, account.getBalance(), "deposit");
            this.saccountRepository.save(account);
            response.setResponseMessage("Rs." + amount + " successfully deposited into your account
balance is now Rs."
                    + account.getBalance());
            response.setDepositStatus(flag);
        } catch (Exception var7) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
            response.setDepositStatus(flag);
        }

        response.setAccount(acc);
        return response;
    }

    public WithdrawResponse withdraw(long acc, int amount) {
        WithdrawResponse response = new WithdrawResponse();
        boolean flag = true;

        try {
            Saccount account = this.saccountRepository.findByAccno(acc);
            User user = this.udao.findByUsername(account.getUsername());
            if (user.getFeatureStatus() != 2 && user.getFeatureStatus() != 3) {
                flag = false;
                response.setResponseMessage("This function is not available for your account");
                response.setWithdrawStatus(flag);
            } else if (account.getBalance() >= amount) {
                account.setBalance(account.getBalance() - amount);
                this.service.addAction(acc, amount, account.getBalance(), "withdraw");
                this.saccountRepository.save(account);
                response.setResponseMessage("Rs." + amount + " successfully withdrawn your account
balance is now Rs."
                        + account.getBalance());
                response.setWithdrawStatus(flag);
            } else {
                flag = false;
                response.setResponseMessage("Insufficient funds to complete the transaction");
                response.setWithdrawStatus(flag);
            }
        } catch (Exception var8) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
```

```
            response.setWithdrawStatus(flag);
        }

        response.setAccount(acc);
        return response;
    }

    public TransferResponse transfer(long saccount, long raccount, int amount) {
        TransferResponse response = new TransferResponse();
        boolean flag = true;

        try {
            Saccount senderAccount = this.saccountRepository.findByAccno(saccount);
            User user;
            if (isprimary(raccount)) {
                Account receiverAccount = this.adao.findByAccno(raccount);
                if (senderAccount.getAccno() != receiverAccount.getAccno()) {
                    if (senderAccount.getBalance() >= amount) {
                        user = this.udao.findByUsername(senderAccount.getUsername());
                        if (user.getFeatureStatus() == 3) {
                            senderAccount.setBalance(senderAccount.getBalance() - amount);
                            receiverAccount.setBalance(receiverAccount.getBalance() + amount);
                            this.tservice.addAction(saccount, raccount, amount);
                            this.saccountRepository.save(senderAccount);
                            this.adao.save(receiverAccount);
                            response.setResponseMessage("Rs." + amount + " successfully transferred to
account "
                                    + receiverAccount.getAccno());
                            response.setTransferStatus(flag);
                        } else {
                            flag = false;
                            response.setResponseMessage("This feature is not available for your
account");
                            response.setTransferStatus(flag);
                        }
                    } else {
                        flag = false;
                        response.setResponseMessage("Insufficient funds to complete the transfer");
                        response.setTransferStatus(flag);
                    }
                } else {
                    flag = false;
                    response.setResponseMessage("sender and recieiver accounts are same");
                    response.setTransferStatus(flag);
                }
            } else {
                Saccount receiverAccount = this.saccountRepository.findByAccno(raccount);
                if (senderAccount.getAccno() != receiverAccount.getAccno()) {
                    if (senderAccount.getBalance() > amount) {
                        user = this.udao.findByUsername(senderAccount.getUsername());
                        if (user.getFeatureStatus() == 3) {
                            senderAccount.setBalance(senderAccount.getBalance() - amount);
                            receiverAccount.setBalance(receiverAccount.getBalance() + amount);
                            this.tservice.addAction(saccount, raccount, amount);
                            this.saccountRepository.save(senderAccount);
                            this.saccountRepository.save(receiverAccount);
                            response.setResponseMessage("Rs." + amount + " successfully transferred to
account "
                                    + receiverAccount.getAccno());
```

```
                            response.setTransferStatus(flag);
                        } else {
                            flag = false;
                            response.setResponseMessage("This function isnt available for the account");
                            response.setTransferStatus(flag);
                        }
                    } else {
                        flag = false;
                        response.setResponseMessage("Insufficient funds to complete the transfer");
                        response.setTransferStatus(flag);
                    }
                } else {
                    flag = false;
                    response.setResponseMessage("sender and recieiver accounts are same");
                    response.setTransferStatus(flag);
                }
            }
        } catch (Exception var11) {
            flag = false;
            response.setResponseMessage("Account number is incorrect");
            response.setTransferStatus(flag);
        }

        response.setSaccount(saccount);
        return response;
    }

    public static boolean isprimary(long account) {
        String s = Long.toString(account).substring(0, 10);
        String check = "3914918201";
        return s.equals(check);
    }

}
```

```
package com.icin.service;

import java.time.LocalDate;
import java.util.List;
import java.util.ArrayList;
import java.util.Collections;

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

import com.icin.entity.Transfer;
import com.icin.repository.TransferHistoryRepository;

@Service
public class TransferHistoryService {

    @Autowired
    private TransferHistoryRepository tHistoryRepository;

    public Transfer addAction(long saccount, long raccount, int amount) {
        LocalDate today = LocalDate.now();
        Transfer transfer = new Transfer();
```

```
        transfer.setSaccount(saccount);
        transfer.setRaccount(raccount);
        transfer.setAmount(amount);
        transfer.setDate(today);
        return (Transfer) this.tHistoryRepository.save(transfer);
    }

    public List<Transfer> getTransfers(long account) {
        List<Transfer> sender = this.tHistoryRepository.findBySaccount(account);
        List<Transfer> receiver = this.tHistoryRepository.findByRaccount(account);
        List<Transfer> merged = new ArrayList<Transfer>();
        merged.addAll(sender);
        merged.addAll(receiver);
        Collections.sort(merged);
        return merged;
    }
}
```

```
package com.icin.service;

import java.time.LocalDate;
import java.util.List;

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

import com.icin.entity.UserHistory;
import com.icin.repository.UserHistoryRepository;

@Service
public class UserHistoryService {

    @Autowired
    private UserHistoryRepository dao;

    public UserHistory addAction(long account, int amount, int balance, String action) {
        LocalDate today = LocalDate.now();
        UserHistory row = new UserHistory();
        row.setAccount(account);
        row.setAction(action);
        row.setAmount(amount);
        row.setDate(today);
        return (UserHistory) this.dao.save(row);
    }

    public List<UserHistory> getHistory(long account) {
        return this.dao.findByAccount(account);
    }

}
```

```
package com.icin.service;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import com.icin.entity.User;
import com.icin.repository.UserRepository;

@Service
public class UserService {

    @Autowired
    private UserRepository userRepo;

    public User findByEmail(String email) {
        return userRepo.findByEmail(email);
    }

    public User findByUsername(String name) {
        return null;
    }

    public User findByFullname(String name) {
        return userRepo.findByUsername(name);
    }

}
```

```
server.port=8080
spring.jpa.hibernate.ddl-auto=update
spring.datasource.url=jdbc:mysql://localhost:3306/icin
spring.datasource.username=root
spring.datasource.password=Redhat@1234.
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.jpa.show-sql = true
```

```
<!-- <p>adminregister works!</p> -->
<div class="container">
    <div>
        <h4 class="fw-bold p-2">Unauthorized Users</h4>
        <div>
            <table class="table table-bordered text-center table-sm">
                <thead>
                    <tr>
                        <th>First Name</th>
                        <th>Last Name</th>
                        <th>Phone Number</th>
                        <th>Address</th>
                        <th>Date of Birth</th>
                        <th>Type of Verification Document</th>
                        <th>Verification ID No.</th>
                        <th>Email</th>
                        <th>Create Account</th>
                        <th>Cancel Account Creation</th>
                    </tr>
                </thead>
                <tbody>
                    <tr *ngFor="let authorizeuser of authorizeusers">
```

```
                        <td>{{authorizeuser.fname}}</td>
                        <td>{{authorizeuser.lname}}</td>
                        <td>{{authorizeuser.phone}}</td>
                        <td>{{authorizeuser.address}}</td>
                        <td>{{authorizeuser.dob}}</td>
                        <td>{{authorizeuser.identityType}}</td>
                        <td>{{authorizeuser.identity}}</td>
                        <td>{{authorizeuser.email}}</td>
                        <td><button type="button" class="btn btn-success"

(click)="authorizeAccount(authorizeuser.username)">Create</button>    </td>
                         <td><button type="button" class="btn btn-danger"

(click)="rejectRequest(authorizeuser.username)">Cancel</button></td>
                    </tr>
                </tbody>
            </table>
        </div>
    </div>
</div>
```

```
import { Component } from '@angular/core';
import { Authorizeuser } from 'src/app/model/authorizeuser';
import { AuthorizationService } from 'src/app/service/authorization.service';

@Component({
  selector: 'app-adminregister',
  templateUrl: './adminregister.component.html',
  styleUrls: ['./adminregister.component.css']
})
export class AdminregisterComponent {

  authorizeusers: Authorizeuser[];

  constructor(
    public authorizeService: AuthorizationService
  ) { }

  ngOnInit() {
    this.authorizeService.getRequestData().subscribe(res => {
      console.log(res);
      this.authorizeusers = res
    });
  }

  authorizeAccount(username: any) {
    this.authorizeService.authorizeAccount(username).subscribe(res => this.ngOnInit());
  }

  rejectRequest(username: any) {
    this.authorizeService.rejectRequest(username).subscribe(res => this.ngOnInit());
  }
```

```
}
```

```
<!-- <p>checkbookrequest works!</p> -->
<h3 class="p-3">Cheque Book Requests</h3>

<div class="ms-4 me-4">
    <table class="table table-bordered text-center" style="overflow-x:auto;">
        <thead>
            <tr>
                <th>Request Number</th>
                <th>Type of Account</th>
                <th>Account Number</th>
                <th>Applied Date</th>
                <th>Number of Pages</th>
                <th> Confirm Request ?</th>
            </tr>
        </thead>
        <tbody>
            <tr *ngFor="let checkbookrequest of checkbookrequests">
                <td>{{checkbookrequest.id}}</td>
                <td>{{checkbookrequest.accType}}</td>
                <td>{{checkbookrequest.account}}</td>
                <td>{{checkbookrequest.date}}</td>
                <td>{{checkbookrequest.no_of_pages}}</td>
                <td><button (click)="confirmRequest(checkbookrequest.account)" class="btn
btn-primary">Confirm
                        Request</button></td>

            </tr>
        </tbody>
    </table>
</div>
```

```
import { Component } from '@angular/core';
import { Checkbookrequest } from 'src/app/model/checkbookrequest';
import { CheckbookService } from 'src/app/service/checkbook.service';

@Component({
  selector: 'app-checkbookrequest',
  templateUrl: './checkbookrequest.component.html',
  styleUrls: ['./checkbookrequest.component.css']
})
export class CheckbookrequestComponent {

  checkbookrequests: Checkbookrequest[];
  term: string;

  constructor(
    public checkbookService: CheckbookService
  ) { }
```

```
  ngOnInit() {
    this.checkbookService.getRequestsData().subscribe(res => {
      this.checkbookrequests = res
    });
  }

  getData() { }

  confirmRequest(account: any) {
    this.checkbookService.confirmCheckbookService(account).subscribe(res =>
this.ngOnInit());
  }

}
```

```
<!-- <p>login works!</p> -->
<br>
<br>
<br>
<br>
<div class="col-md-4 m-auto">
    <div class="card shadow-lg rounded-0">
        <div class="card-header text-center">
            <h1>Admin Portal</h1>
        </div>
        <div class="card-body">
            <form class="form-login" (ngSubmit)="onSubmit(loginForm)"
#loginForm="ngForm">

                <label for="inputUserName" class="sr-only">UserName</label>
                <input type="text" id="inputUserName" name="inputUserName"
class="form-control"
                    [ngClass]="{ 'is-invalid': submitted && inputUserName.errors }"
ngModel placeholder="UserName"
                    required autofocus #inputUserName=ngModel>
                <div *ngIf="submitted && inputUserName.errors" class="invalid-feedback">
                    <span *ngIf="inputUserName.errors['required']">Please enter a
UserName</span>
                </div>

                <br>
                <label for="inputPassword" class="sr-only">Password</label>
                <input type="password" id="inputPassword" name="password"
class="form-control"
                    [ngClass]="{ 'is-invalid': submitted && password.errors }" ngModel
placeholder="Password" required
                    #password=ngModel>
                <div *ngIf="submitted && password.errors" class="invalid-feedback">
                    <span *ngIf="password.errors['required']">Please enter a
Password</span>
                </div>
```

```
                <br>
                <button class="btn btn-primary ps-4 pe-4" type="submit">Login</button>
            </form>
        </div>
    </div>
</div>
```

```
import { Component } from '@angular/core';
import { NgForm } from '@angular/forms';
import { AuthenticationService } from 'src/app/service/authentication.service';
import { Logindata } from 'src/app/model/logindata';

@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent {

  constructor(
    private authenticationService: AuthenticationService
  ) { }

  submitted = false;

  ngOnInit(): void {
  }

  onSubmit(loginForm: NgForm) {
    this.submitted = true;
    if (loginForm.valid) {
      const loginDataInstance = new Logindata(loginForm.value.inputUserName,
loginForm.value.password);
      this.authenticationService.authenticate(loginDataInstance);
    }
  }

}
```

```
<!-- <p>useraccount works!</p> -->

<div class="container">
    <h3 class="p-2 bg-secondary-subtle">User Accounts</h3>
    <table class="table table-bordered text-capitalize table-sm table-hover text-center">
        <thead>
            <tr>
                <th>First Name</th>
                <th>Last Name</th>
                <th>UserName</th>
                <!--<th>Email</th>-->
```

```
                <th>Primary Account Number</th>
                <th>Savings Account Number</th>
                <th>Enable Account</th>
                <th>Disable Account</th>
                <th>Features Granted to the User</th>
                <th>Change Features</th>
            </tr>
        </thead>
        <tbody>
            <tr *ngFor="let user of users">
                <td>{{user.fname}}</td>
                <td>{{user.lname}}</td>
                <td>{{user.username}}</td>
                <td>{{user.primaryAccno}}</td>
                <td>{{user.savingsAccno}}</td>
                <td><button  (click)="enableLoginService(user.username)"
                        class="btn btn-success btn-sm rounded-pill">Enable</button></td>
                <td><button  (click)="disableLoginService(user.username)"
                        class="btn btn-danger btn-sm rounded-pill">Disable</button></td>
                <td>{{user.featureStatus}}</td>
                <td>
                    <!-- <select (change)="filterSelected($event.target.value)">
                        <option *ngFor="let r of roles" [value]='r.value'
[selected]="r.value === 0">
                            {{r.name}}
                        </option>
                    </select> -->
                    <div class="btn-group">
                        <select (change)="filterSelected($event)" class="form-select
border-success form-select-sm">
                            <option *ngFor="let r of roles" [value]='r.value'>
                                {{r.name}}
                            </option>
                        </select>
                        <button (click)="setOption(user.username)" class="btn btn-success
btn-sm">Set</button>
                    </div>

                </td>
            </tr>
        </tbody>
    </table>
</div>
```

```
import { Component } from '@angular/core';
import { DisableService } from 'src/app/service/disable.service';
import { EnableService } from 'src/app/service/enable.service';
import { AuthorizationService } from 'src/app/service/authorization.service';
import { Userdata } from 'src/app/model/userdata';
import { Authorizeuser } from 'src/app/model/authorizeuser';
import { FeaturesService } from 'src/app/service/features.service';
```

```
@Component({
  selector: 'app-useraccount',
  templateUrl: './useraccount.component.html',
  styleUrls: ['./useraccount.component.css']
})
export class UseraccountComponent {

  users: any[];
  selectedOption: string;
  selectedValue: number;

  roles = [
    { name: " ", value: 0 },
    { name: "Deposit Access", value: 1 },
    { name: "Deposit + Withdraw Access", value: 2 },
    { name: "Deposit + Withdraw + Transfer Access", value: 3 }
  ]

  constructor(
    private enable: EnableService,
    private disable: DisableService,
    private authservice: AuthorizationService,
    private featureService: FeaturesService
  ) { }

  ngOnInit(): void {
    this.authservice.getAllUsers().subscribe(res => {
      console.log(res)
      //console.log(res[0].featureStatus)
      res.forEach(element => {

        console.log(element.featureStatus)
        // console.log(this.roles[1].name)
        if (element.featureStatus == 1) {
          element.featureStatus  =  this.roles[1].name
        }
        if (element.featureStatus == 2) {
          element.featureStatus = this.roles[2].name
        }
        if (element.featureStatus == 3) {
          element.featureStatus = this.roles[3].name
        }
      });
      this.users = res

    });
  }

  getAllUsers() {
    this.authservice.getRequestData().subscribe(
      res => {
```

```
        console.log(res);
        this.users = JSON.parse(JSON.stringify(res));
      },
      error => console.log(error)
    )
  }


  enableLoginService(username: string) {
    console.log(username)
    this.enable.enableLoginService(username).subscribe(res => this.ngOnInit());
    //this.enableService.enableLoginService();
  }

  disableLoginService(username: string) {
    this.disable.disableLoginService(username).subscribe(res => this.ngOnInit());
    //this.disableService.disableLoginService();
  }

  setOption(username: string) {
    this.featureService.setFeatures(username, this.selectedValue).subscribe(res =>
this.ngOnInit());

    //this.featuresService.setFeatures();
  }

  filterSelected(event: Event) {
    const selectElement = event.target as HTMLSelectElement;
    const selectedValue = selectElement.value;
    console.log('Selected value:', selectedValue);
  }

}
```

```
import { ActivatedRouteSnapshot, CanActivateFn, Router, RouterStateSnapshot, UrlTree }
from '@angular/router';
import { AuthenticationService } from '../service/authentication.service';
import { Observable } from 'rxjs';

export const authGuard: CanActivateFn = (route, state) => {
  return true;
};


export class AuthGuard {

  constructor(
    private authenticationService: AuthenticationService,
    private router: Router
  ) {

  }
```

```
  canActivate(
    next: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean |
UrlTree> | boolean | UrlTree {
    if (this.authenticationService.isAuthenticated) {
      return true;
    }
    this.router.navigate(['']);
    return false;
  }

}
```

```
export class Authorizeuser {
    fname: string;
    lname: string;
    phone: number;
    address: string;
    pan: string;
    email: string;
    username: string;
    dob: Date;
    identityType: string;
    identity: string;
}
```

```
export class Checkbookrequest {
    id: number;
    accType: string;
    username: string;
    account: number;
    date: string;
    no_of_pages: number;
}
```

```
export class Logindata {
    constructor(
        public username: string,
        public password: string
    ) { }
}
```

```
export class Userdata {
    fname: string;
    lname: string;
    username: string;
    email: string;
    primaryAccno: number;
    savingsAccno: number;
```

```
    featureStatus: boolean;
}
```

```
import { Injectable } from '@angular/core';
import { Logindata } from '../model/logindata';
import { Router } from '@angular/router';

@Injectable({
  providedIn: 'root'
})
export class AuthenticationService {

  private readonly adminUser = new Logindata('Uzair', 'Uzai');
  isAuthenticated = false;
  constructor(private router: Router) { }

  authenticate(login: Logindata): boolean {
    if (this.checkCredentials(login)) {
      this.isAuthenticated = true;
      this.router.navigate(['../useraccount']);
      return true;
    }
    alert("Incorrect Login or Password");
    this.router.navigate(['../']);
    this.isAuthenticated = false;
    return false;
  }

  checkCredentials(login: Logindata): boolean {
    return this.checkEmail(login.username) && this.checkPassword(login.password);
  }

  checkEmail(email: string): boolean {
    return email === this.adminUser.username;
  }

  checkPassword(password: string): boolean {
    return password === this.adminUser.password;
  }

  logout() {
    this.isAuthenticated = false;
    this.router.navigate(['']);
  }

}
```

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http'
import { Authorizeuser } from '../model/authorizeuser';
import { Observable } from 'rxjs';
```

```
import { Userdata } from '../model/userdata';


@Injectable({
  providedIn: 'root'
})
export class AuthorizationService {

  private rootUrl = 'http://localhost:8080/user';

  constructor(private http: HttpClient) { }

  getAllUsers() {
    return this.http.get<any[]>(this.rootUrl + '/all');
  }

  getRequestData() {
    return this.http.get<any[]>(this.rootUrl + '/unauthorized/all');
  }

  authorizeAccount(username: Authorizeuser) {
    return this.http.get(this.rootUrl + username + '/authorize');
  }

  rejectRequest(username: Authorizeuser) {
    return this.http.get(this.rootUrl + username + '/authorize/cancel');
  }

}
```

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Checkbookrequest } from '../model/checkbookrequest';

@Injectable({
  providedIn: 'root'
})
export class CheckbookService {

  readonly rootUrl = 'http://localhost:8080/user/';

  readonly dataUrl = 'http://localhost:8080/chequebook/request/all';

  private data: any = []

  constructor(private http: HttpClient) { }

  confirmCheckbookService(account: number) {
    return this.http.get(this.rootUrl + account + '/chequebook/request/confirm');
  }
```

```
  getRequestsData(): Observable<Checkbookrequest[]> {
    return this.http.get<Checkbookrequest[]>(this.dataUrl);
  }

}
```

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class DisableService {

  readonly rootUrl = 'http://localhost:8080/user/';

  constructor(private http: HttpClient) { }

  disableLoginService(username: any) {
    return this.http.get(this.rootUrl + username + '/disable');
  }

}
```

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class EnableService {

  readonly rootUrl = 'http://localhost:8080/user/';

  constructor(private http: HttpClient) { }

  enableLoginService(username: any) {
    return this.http.get(this.rootUrl + username + '/enable');
  }

}
```

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable, throwError } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class FeaturesService {
```

```
  id: number

  readonly rootUrl = 'http://localhost:8080/user/';

  constructor(private http: HttpClient) { }

  setFeatures(username: string, value: number | string): Observable<any> {
    // Ensure that parameters are defined
    if (!username || value === undefined || value === null) {
      console.error('Invalid parameters!', { username, value });
      return throwError('Invalid parameters!');
    }
    const url = `${this.rootUrl}${username}/features/${value}`;
    return this.http.get(url);
  }
}
```

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { LoginComponent } from './components/login/login.component';
import { UseraccountComponent } from './components/useraccount/useraccount.component';
import { authGuard } from './guard/auth.guard';
import { CheckbookrequestComponent } from
'./components/checkbookrequest/checkbookrequest.component';
import { AdminregisterComponent } from
'./components/adminregister/adminregister.component';

const routes: Routes = [
  { path: '', component: LoginComponent },
  { path: 'useraccount', component: UseraccountComponent, canActivate: [authGuard] },
  { path: 'ckeckbook', component: CheckbookrequestComponent, canActivate: [authGuard] },
  { path: 'authorize', component: AdminregisterComponent, canActivate: [authGuard] },
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

```
<div  *ngIf="authenticationService.isAuthenticated">
  <nav class="navbar navbar-expand-sm navbar-dark bg-dark">
    <a class="navbar-brand" routerLink="/user-account">ICIN Bank</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarsExample02"
      aria-controls="navbarsExample02" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
```

```
    <div class="collapse navbar-collapse" id="navbarsExample02">
    <ul class="navbar-nav mr-auto">

                    <li class="nav-item">
                      <a class="nav-link"  routerLink="/useraccount">User Account Info</a>
          </li>
          <li class="nav-item">
          <a class="nav-link"                routerLink="/ckeckbook">CheckBook Requests</a>
          </li>
          <li class="nav-item">
          <a class="nav-link"                routerLink="/authorize">Authorization(KYC)</a>
          </li>
          <li class="nav-item">
          <a class="nav-link"


</ul>
</div>
</nav>
</div>
<router-outlet></router-outlet>
```

```
import { Component } from '@angular/core';
import { AuthenticationService } from './service/authentication.service';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'ICINBank';

  constructor(
    public authenticationService: AuthenticationService
  ) { }

  logout() {
    this.authenticationService.logout();
  }
}
```

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { LoginComponent } from './components/login/login.component';
import { UseraccountComponent } from './components/useraccount/useraccount.component';
```

```
import { CheckbookrequestComponent } from './components/checkbookrequest/checkbookrequest.component'; import {
AdminregisterComponent } from './components/adminregister/adminregister.component';

import { HttpClientModule } from '@angular/common/http' import { FormsModule } from '@angular/forms';
import { NgbModule } from '@ng-bootstrap/ng-bootstrap';

@NgModule({ declarations: [
AppComponent, LoginComponent, UseraccountComponent, CheckbookrequestComponent, AdminregisterComponent
],
imports: [ BrowserModule, AppRoutingModule, FormsModule, HttpClientModule, NgbModule
],
providers: [],
bootstrap: [AppComponent]
})
export class AppModule { }
```