PROJECT REPORT

ON

Bank Management System (Online Banking)

B.Tech (CE) Sem-IV

In the Subject of

SEPP & SP

Gohel Chandresh Shaileshkumar (CE-039) (19CEUOS128)

Gopani Kenil Ghanshyambhai (CE-040) (19CEUOS080)

Under the Guidance of

Prof. Brijesh Bhatt
Prof. Jigar Pandya
Prof. Pinkal Chauhan

Department Of Computer Engineering

Contents

1. Introduction/Abstract……………………………………3
2. Software Requirement Specifications…………………4
3. Design………………………………………………….…7
4. Implementation Detail…………………………….….18
5. Work Flow/Layouts ……………………………………26
6. Conclusion…………………………………………….36
7. Limitation and Future Extension……………………..37
8. Bibliography…………………………………………..38

1. Introduction

This project is clone online banking system with minimal number major functionalities.

Project created using Django and other tools such that it supports features which are provided by other online banking software.

This project system works for same self-bank speedy transfer, check transactions, update profiles and raise an issue.

When users creates account they will be notified by email which contains their account number and IFSC code, after sending mail system redirects to register for online banking page. After success fully registration user can login into the system. There is responsibility of admin to add initial amount to the account manually at bank.

There is four option user can choose “View Balance”, “Make Transaction”, “Transaction History”, “Contact Us”.

By choosing “Contact Us” users can raise an issue to admin what they facing.

“Make Transaction” works for transfer money from one account to another after authentication, both user gets mail of that transaction. “View Balance” works for viewing amount in user account, and “Transaction History” works for seeing all the transaction user had made. “Logout” button is there to gets logout at any time.

2. Software Requirement Specifications

Users of Software:

1. Bank account holder
2. Bank Staff
3. Admin

1. Manage Account:

Description: User can create, delete and update account by managing it.

State:-Users are at home page, some option which is displayed for doing particular task.

R 1.1 Create account.

Input: "Create Account" option selection

Output: Form will be displayed which required some details.

R 1.1.1 Enter details.

Description: Here, users have to fill some personal details like name, age, adhaar-card number, birthdate, and address for creating an account in bank.
Input: Enter asked details of user in the given form and submit the form.
Output: An email goes to user mail and account number and IFSC code will be there.

R 1.2 Delete account.

Input: "Delete an account" option selection.
Output: Form will be displayed which required some details.

R 1.2.1 Enter details.

Description: User need to provide details about their account and have to submit it. Then after some days account will be deleted successfully.

Input: Enter details about user account as per asked in form and submit the form.

Output: conformation message will be send after successfully deletion

2. Manage User:

R 2.1 Register into online banking services

Description: At time of registration, it’s necessary that users have to enter required information asked in form. Once this step done, then in future no need enter all those details only user-id and password is sufficient.
State: Form with asking of required information like account holder name, account number, IFS-Code of bank, user-name (user choice) as ID and password.
Output: Registered successfully redirect to Login page.

R 2.2 User login
Input: Enter user-id, password.
Output: You are success fully logged in (Redirect to home page).
(If not matched with data base stayed on login page).
R 2.3 Update own information.
Description: A person who changed his/her residential address, mobile no., email address; after logged into their account they can modify their information securely by selecting this option.
Input: Select option which you want to update.
(I.e. Mobile number, Residential address, Email address)
Output: Your information is updated.

3. Manage Transaction:
R 3.1 Transfer Money:
Description: A person can transfer money securely from his/her account to another’s account by selecting this option. Person will be asked for verification code for transaction.

R 3.1.1 Enter account detail where to transfer money
State: A dialog box for asking beneficiary name, account number where to transfer money.
Input: Enter that beneficiary account details.
R 3.1.2 Enter amount of money to transfer
State: A dialog box for asking amount of money to transfer.
Input: Enter amount and press confirm.
R 3.1.3 Check for sufficient amount in account:
Description: To check that person has sufficient amount in his/her account for debit.

(If not sufficient amount of money in account)

Output: No sufficient amount of money in account. Aborting transaction.

(**If have sufficient amount of money OTP will be sent)**

State: Confirmation for transaction

R 3.2 Confirmation for transfer money:

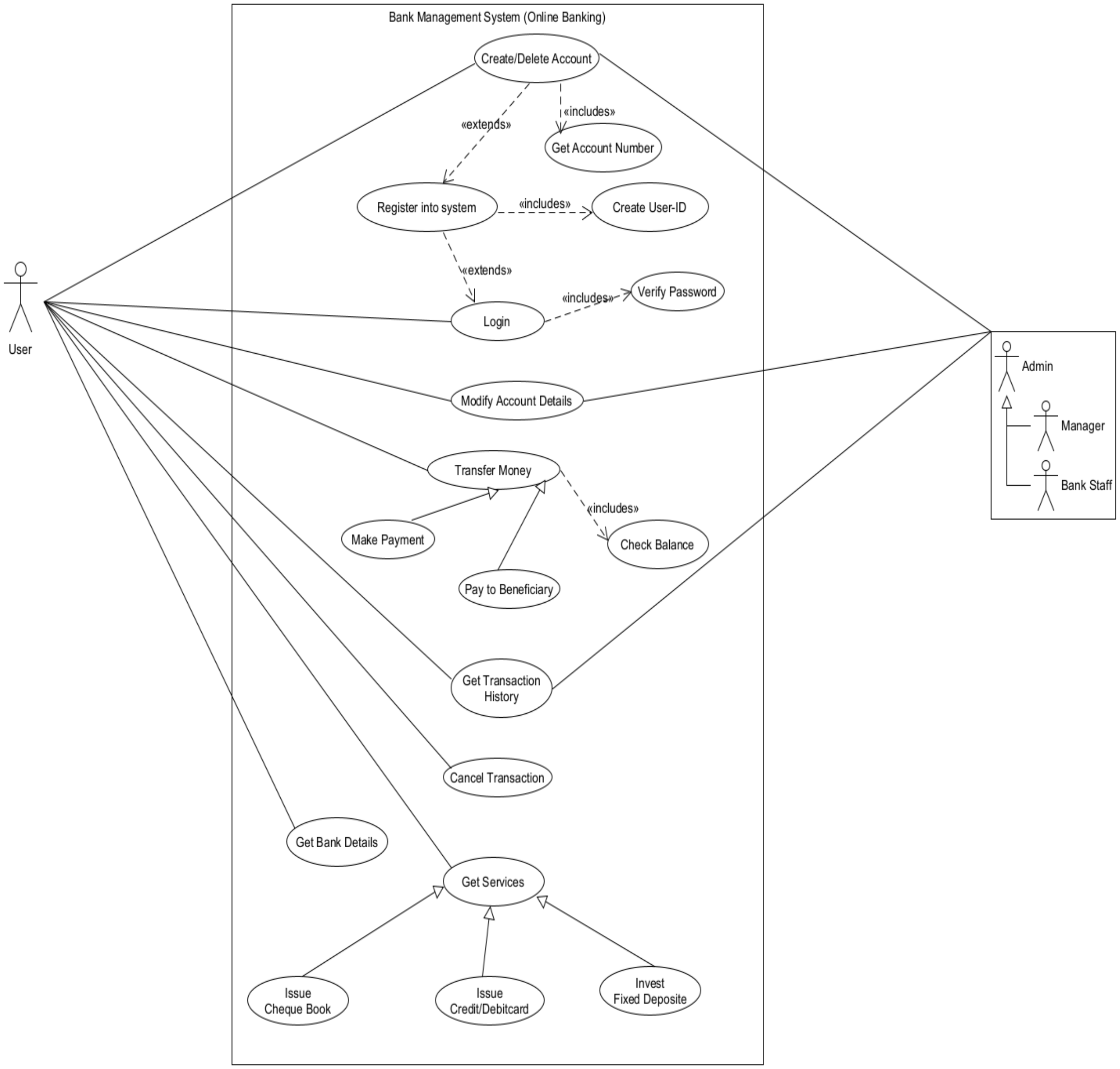
State: A dialog box asking for OTP delivered to email and login password.
Input: Enter OTP and login password.
Output: Your transaction done successfully.

R 3.4 Transaction History

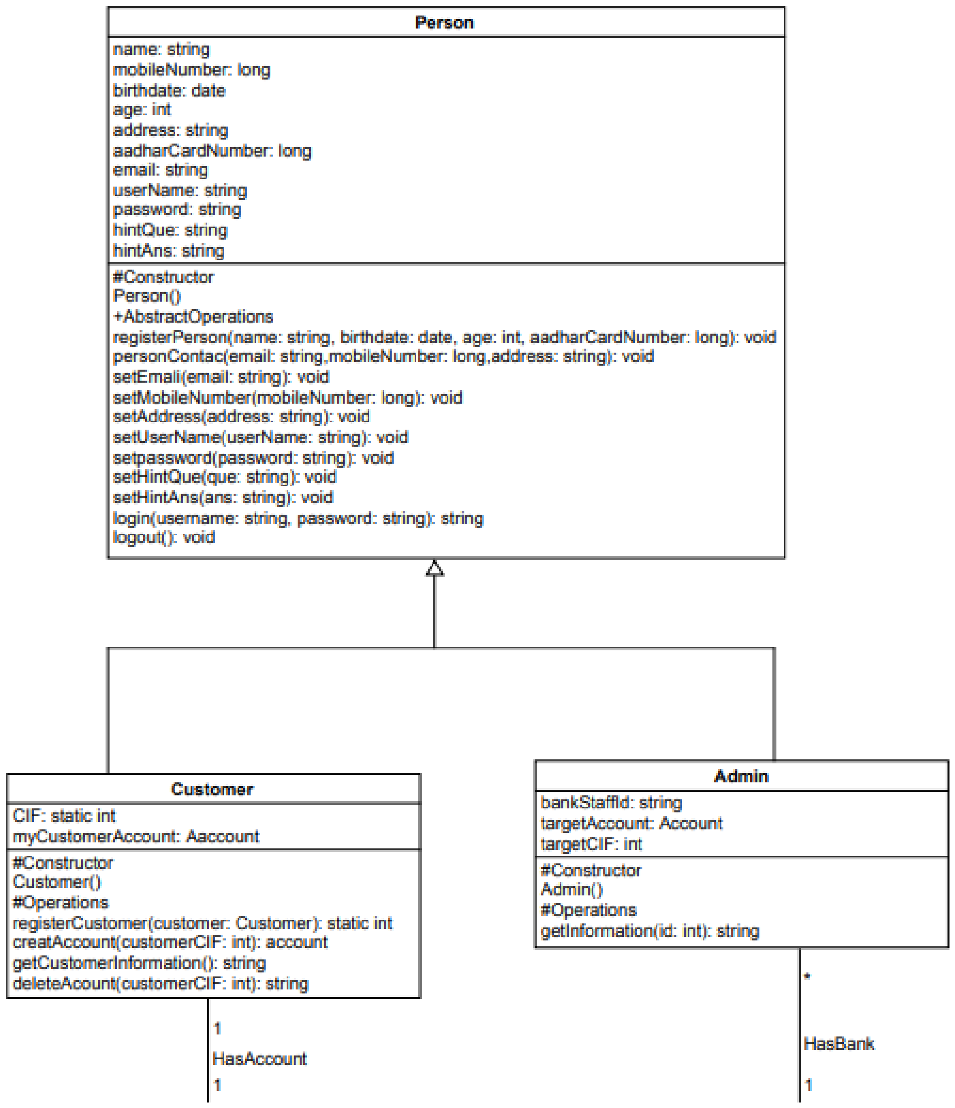
Description: Person can see his/her account transaction history.
Input: Click on button to see transaction history.
Output: Page containing transaction with full details.

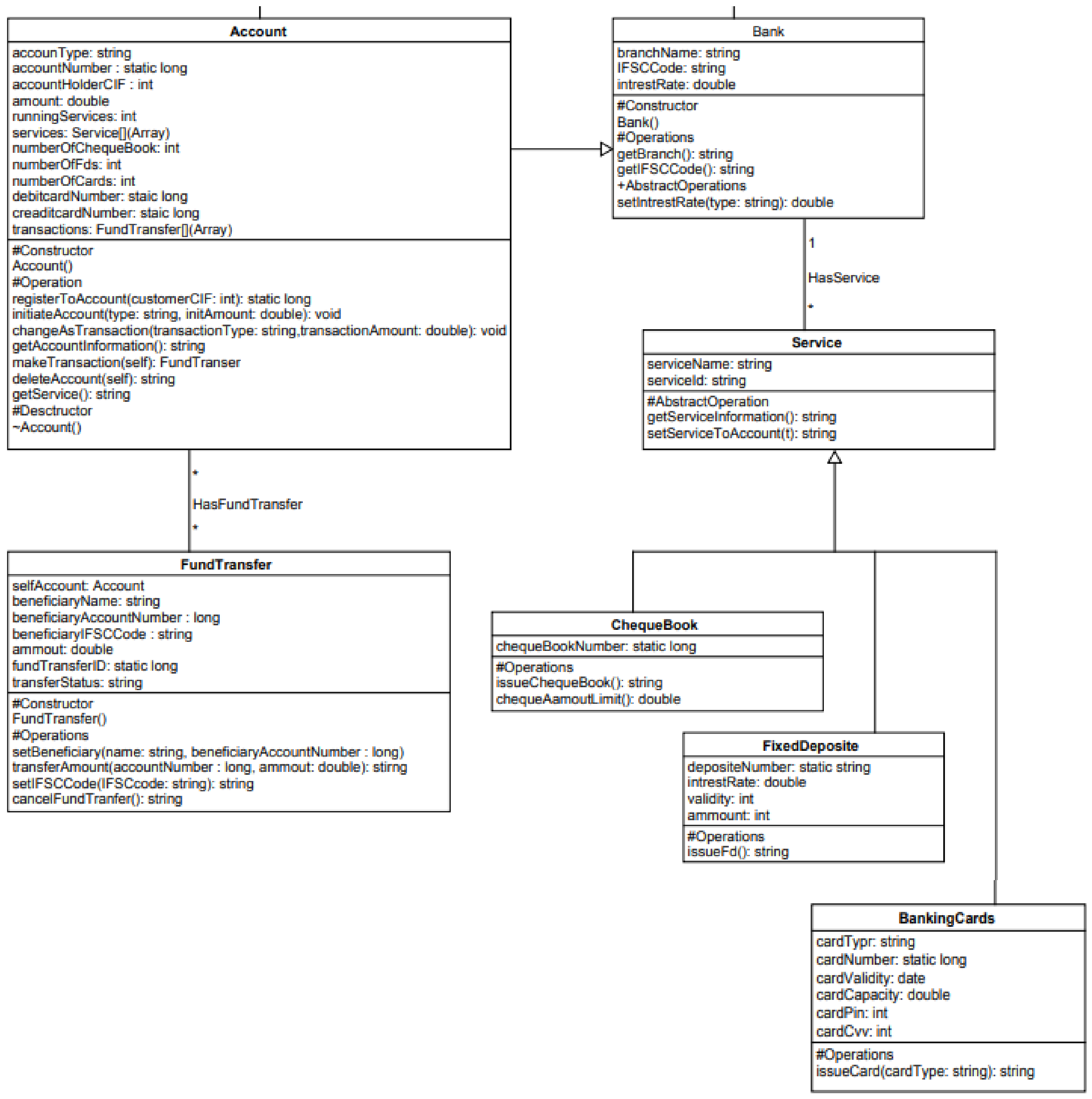
3. Designs Documents

1. Use case Diagram:



2. Class Diagram

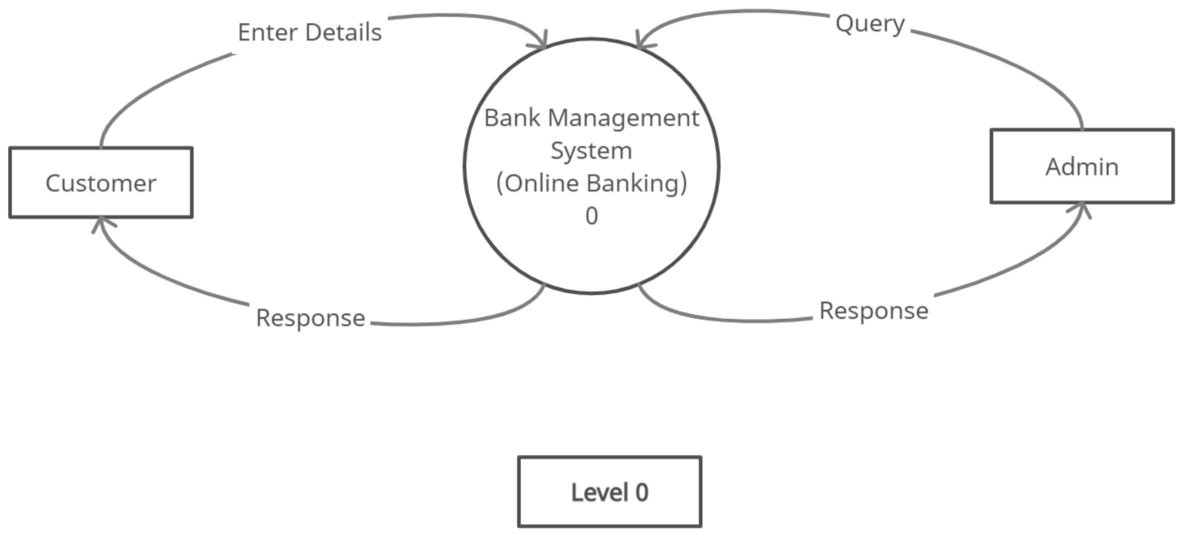

Person
name: string
mobileNumber: long
birthdate: date
age: int
address: string
aadharCardNumber: long
email: string
userName: string
password: string
hintQue: string
hintAns: string
#Constructor
Person()
+AbstractOperations
registerPerson(name: string, birthdate: date, age: int, aadharCardNumber: long): void
personContac(email: string,mobileNumber: long,address: string): void
setEmali(email: string): void
setMobileNumber(mobileNumber: long): void
setAddress(address: string): void
setUserName(userName: string): void
setpassword(password: string): void
setHintQue(que: string): void
setHintAns(ans: string): void
login(username: string, password: string): string
logout(): void
Customer
CIF: static int
myCustomerAccount: Aaccount
#Constructor
Customer()
#Operations
registerCustomer(customer: Customer): static int
creatAccount(customerCIF: int): account
getCustomerInformation(): string
deleteAcount(customerCIF: int): string
Admin
bankStaffId: string
targetAccount: Account
targetCIF: int
#Constructor
Admin()
#Operations
getInformation(id: int): string
1
HasAccount
1
*
HasBank
1


Account
accounType: string
accountNumber : static long
accountHolderCIF : int
amount: double
runningServices: int
services: Service[](Array)
numberOfChequeBook: int
numberOfFds: int
numberOfCards: int
debitcardNumber: staic long
creaditcardNumber: staic long
transactions: FundTransfer[](Array)
#Constructor
Account()
#Operation
registerToAccount(customerCIF: int): static long
initiateAccount(type: string, initAmount: double): void
changeAsTransaction(transactionType: string,transactionAmount: double): void
getAccountInformation(): string
makeTransaction(self): FundTranser
deleteAccount(self): string
getService(): string
#Desctructor
~Account()
Bank
branchName: string
IFSCCode: string
intrestRate: double
#Constructor
Bank()
#Operations
getBranch(): string
getIFSCCode(): string
+AbstractOperations
setIntrestRate(type: string): double
1
HasService
*
Service
serviceName: string
serviceId: string
#AbstractOperation
getServiceInformation(): string
setServiceToAccount(t): string
*
HasFundTransfer
*
FundTransfer
selfAccount: Account
beneficiaryName: string
beneficiaryAccountNumber : long
beneficiaryIFSCCode : string
ammout: double
fundTransferID: static long
transferStatus: string
#Constructor
FundTransfer()
#Operations
setBeneficiary(name: string, beneficiaryAccountNumber : long)
transferAmount(accountNumber : long, ammout: double): stirng
setIFSCCode(IFSCcode: string): string
cancelFundTranfer(): string
ChequeBook
chequeBookNumber: static long
#Operations
issueChequeBook(): string
chequeAamoutLimit(): double
FixedDeposite
depositeNumber: static string
intrestRate: double
validity: int
ammount: int
#Operations
issueFd(): string
BankingCards
cardTypr: string
cardNumber: static long
cardValidity: date
cardCapacity: double
cardPin: int
cardCvv: int
#Operations
issueCard(cardType: string): string

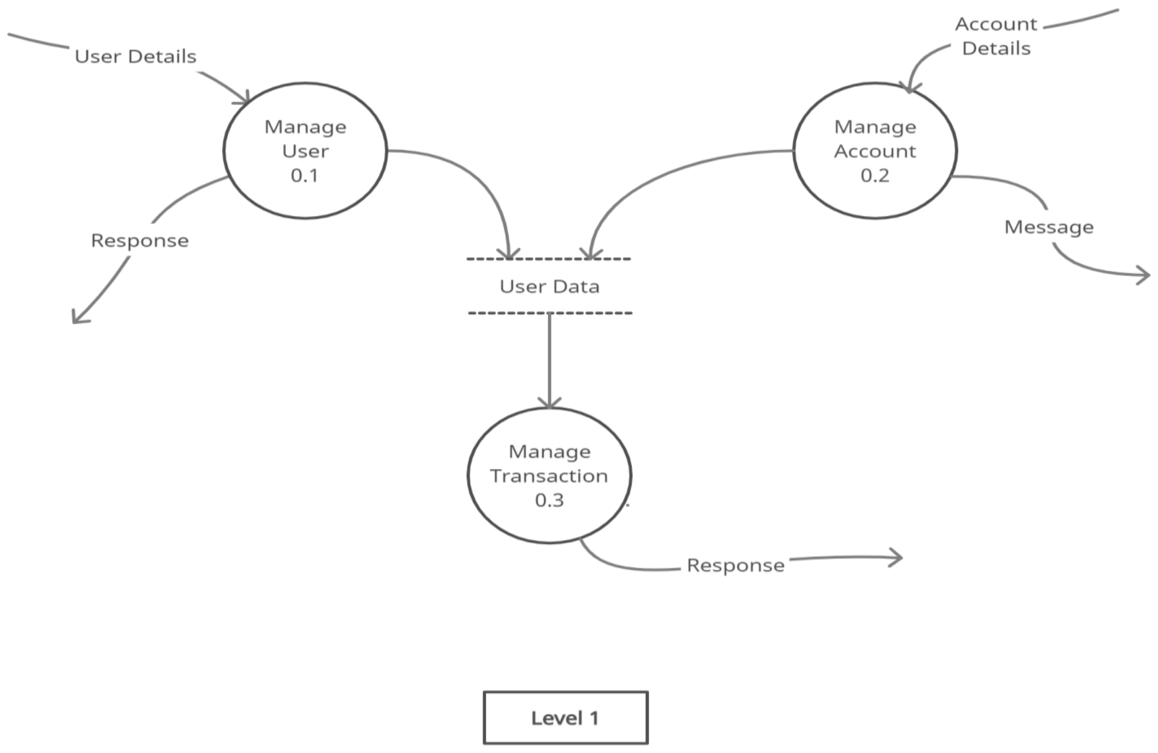
3. DFD Model and Structure Chart:

DFD:

Level: 0

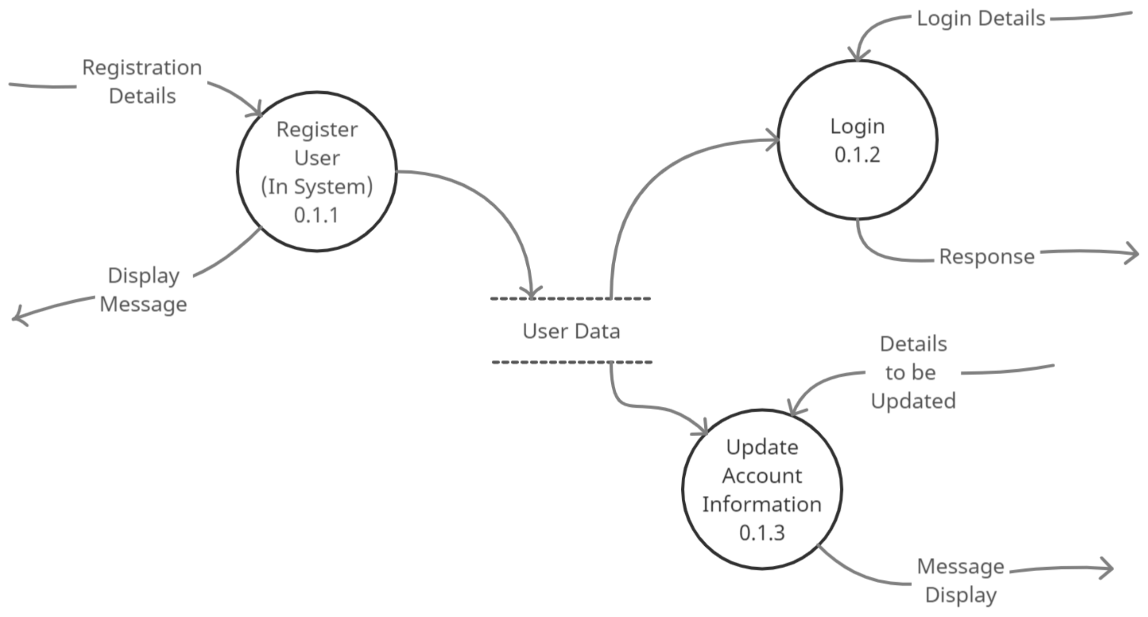


Level: 1

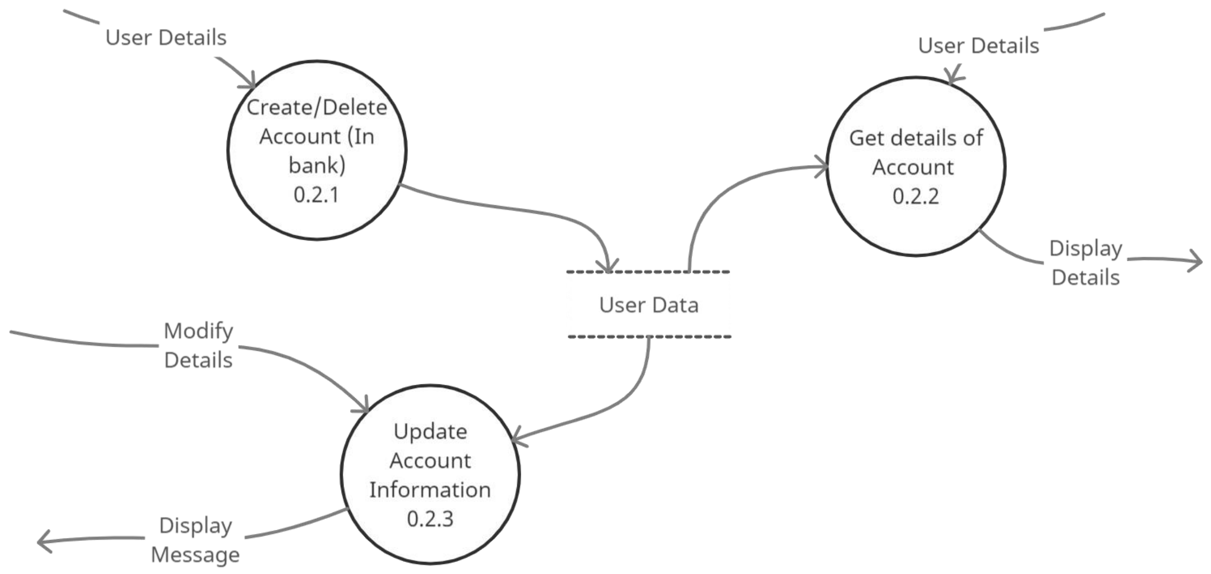


Level: 2

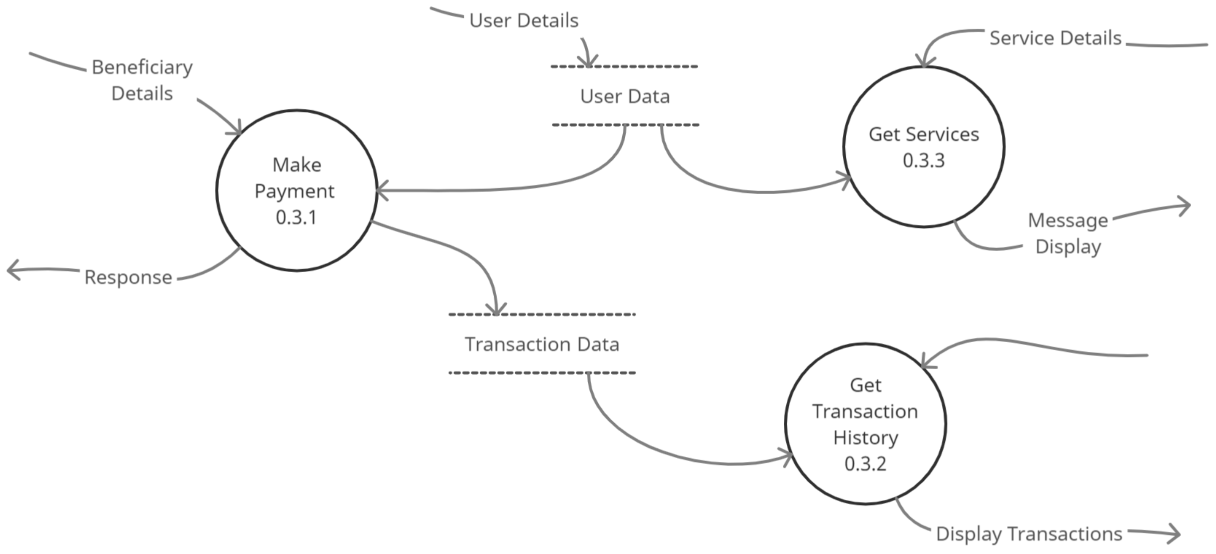
Decompose 0.1



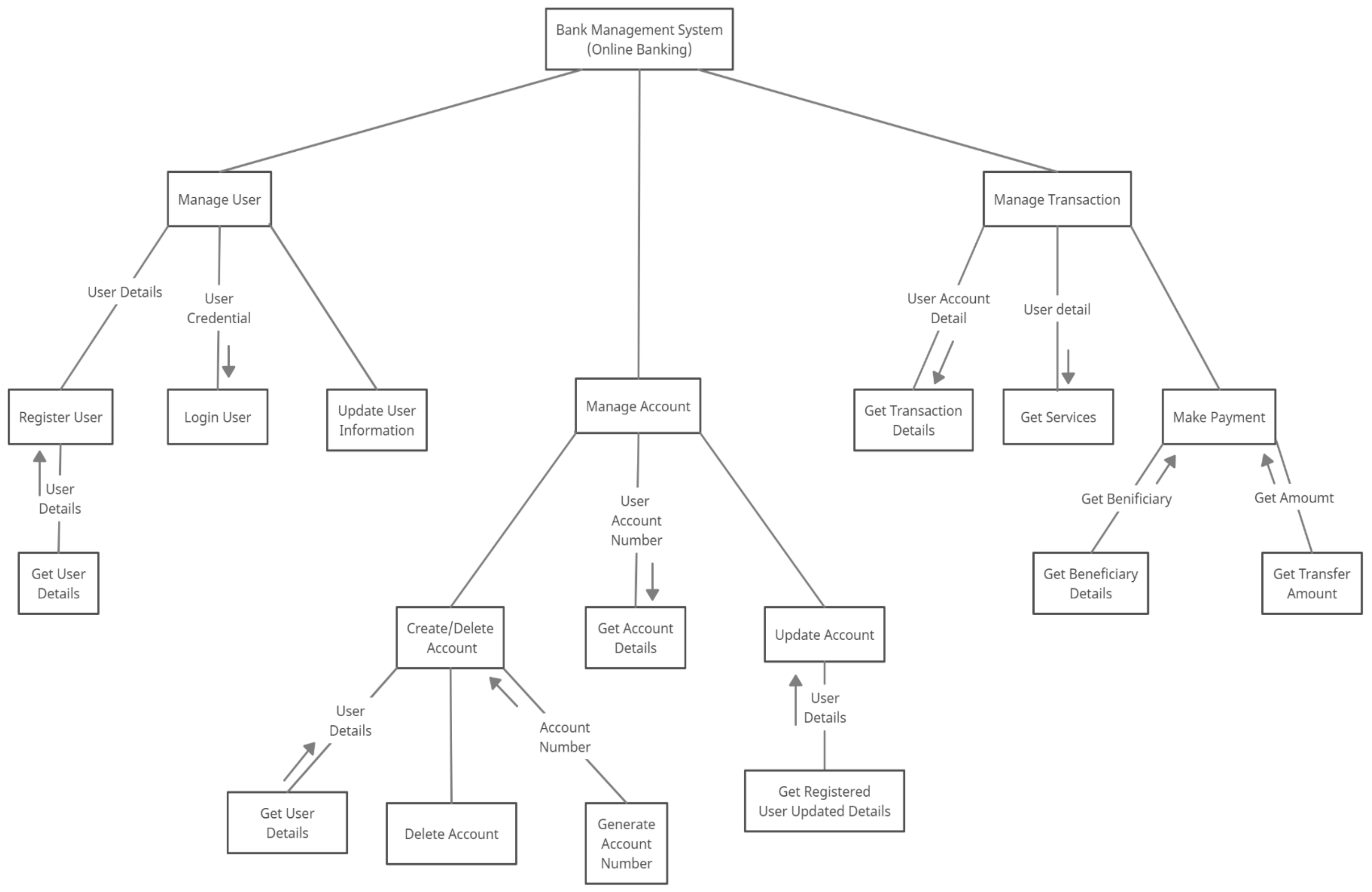
Decompose 0.2


User Details
Create/Delete Account (In bank) 0.2.1
User Data
User Details
Get details of Account 0.2.2
Display Details
Modify Details
Update Account Information 0.2.3
Display Message

Decompose 0.3



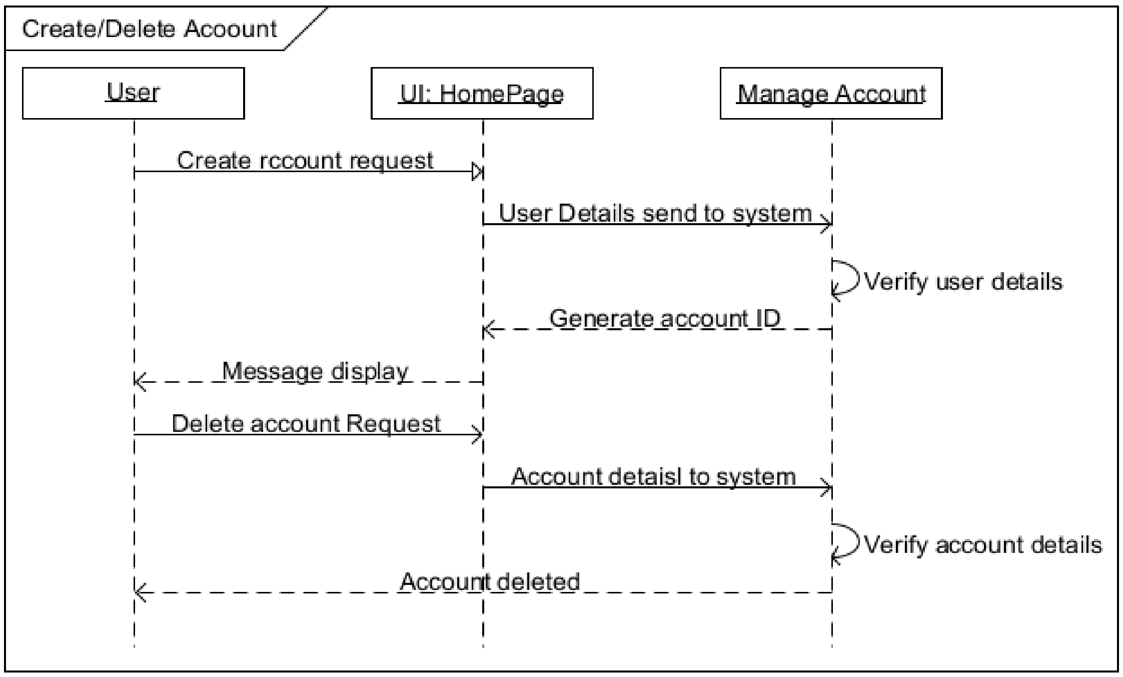
Structure Chart



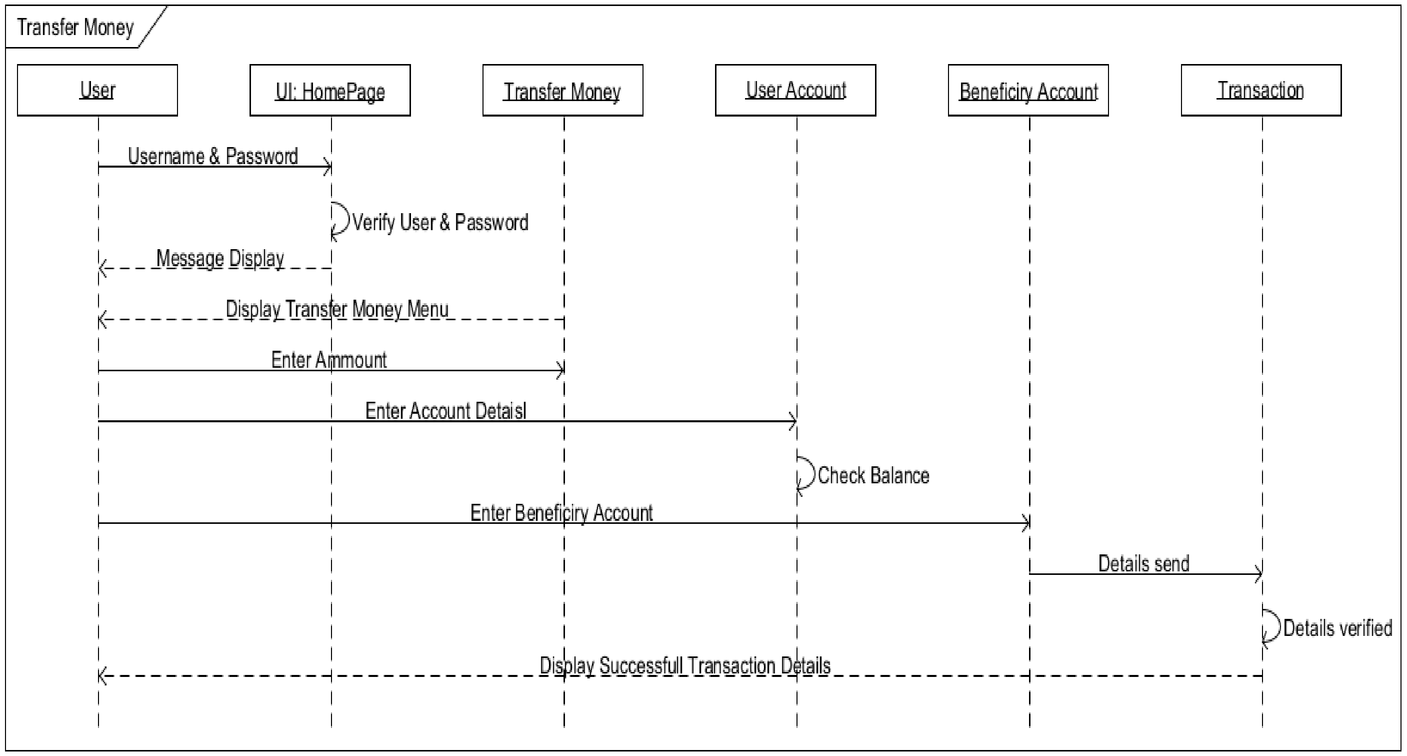
4. Sequence and Activity Diagram:

Sequence Diagram:

Create – Delete Account

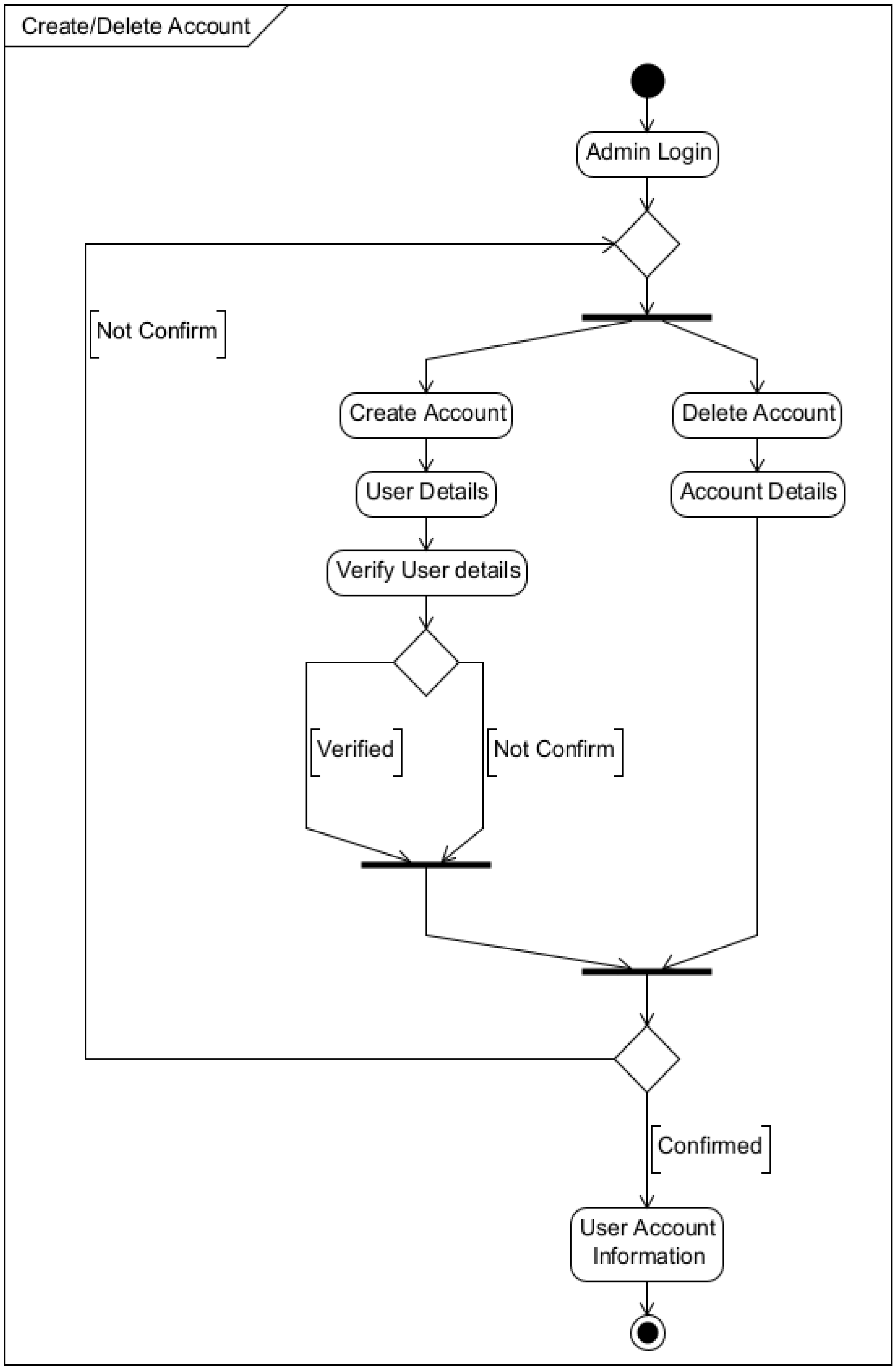


Transfer Money

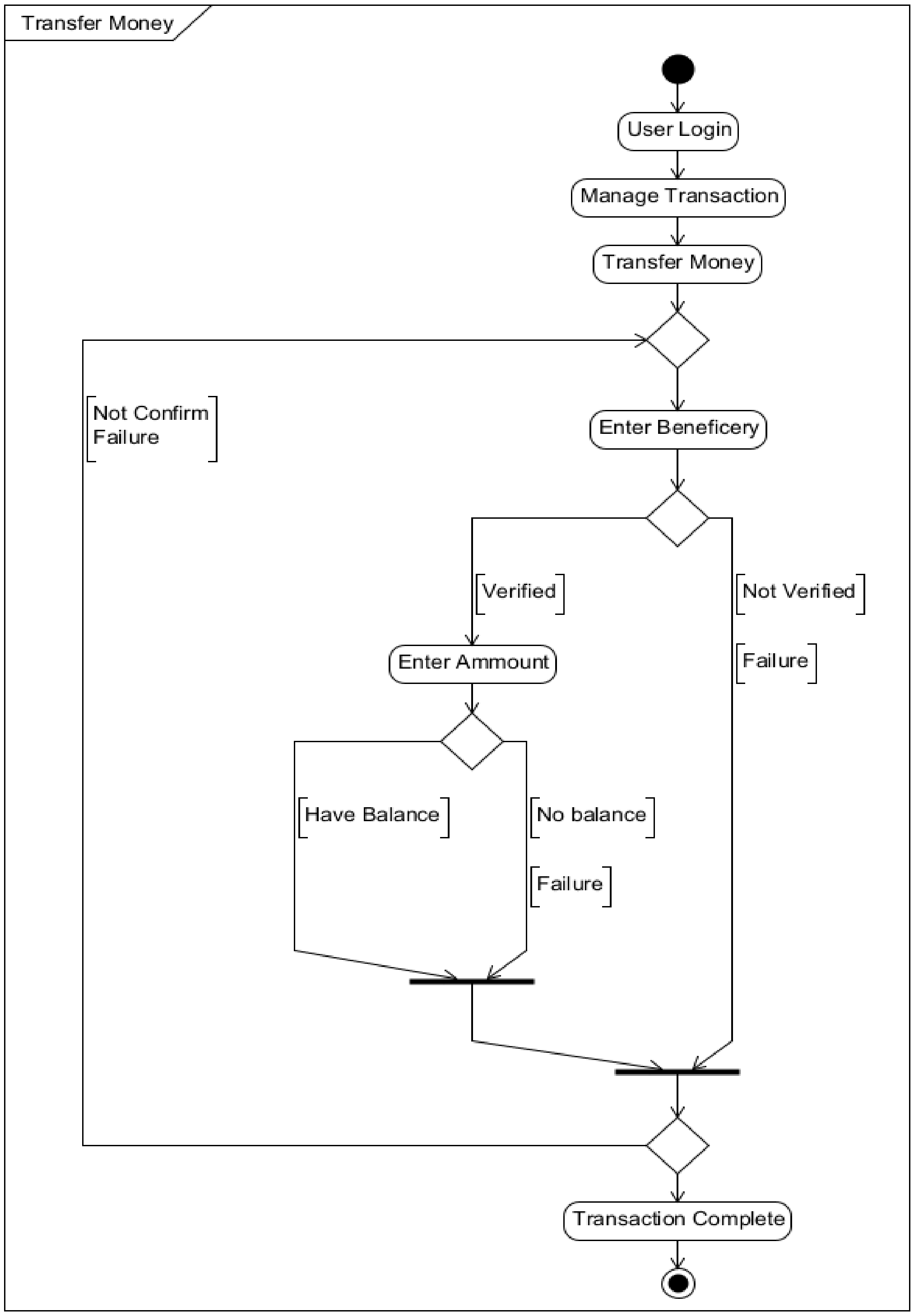


Activity Diagram

Create – Delete Account

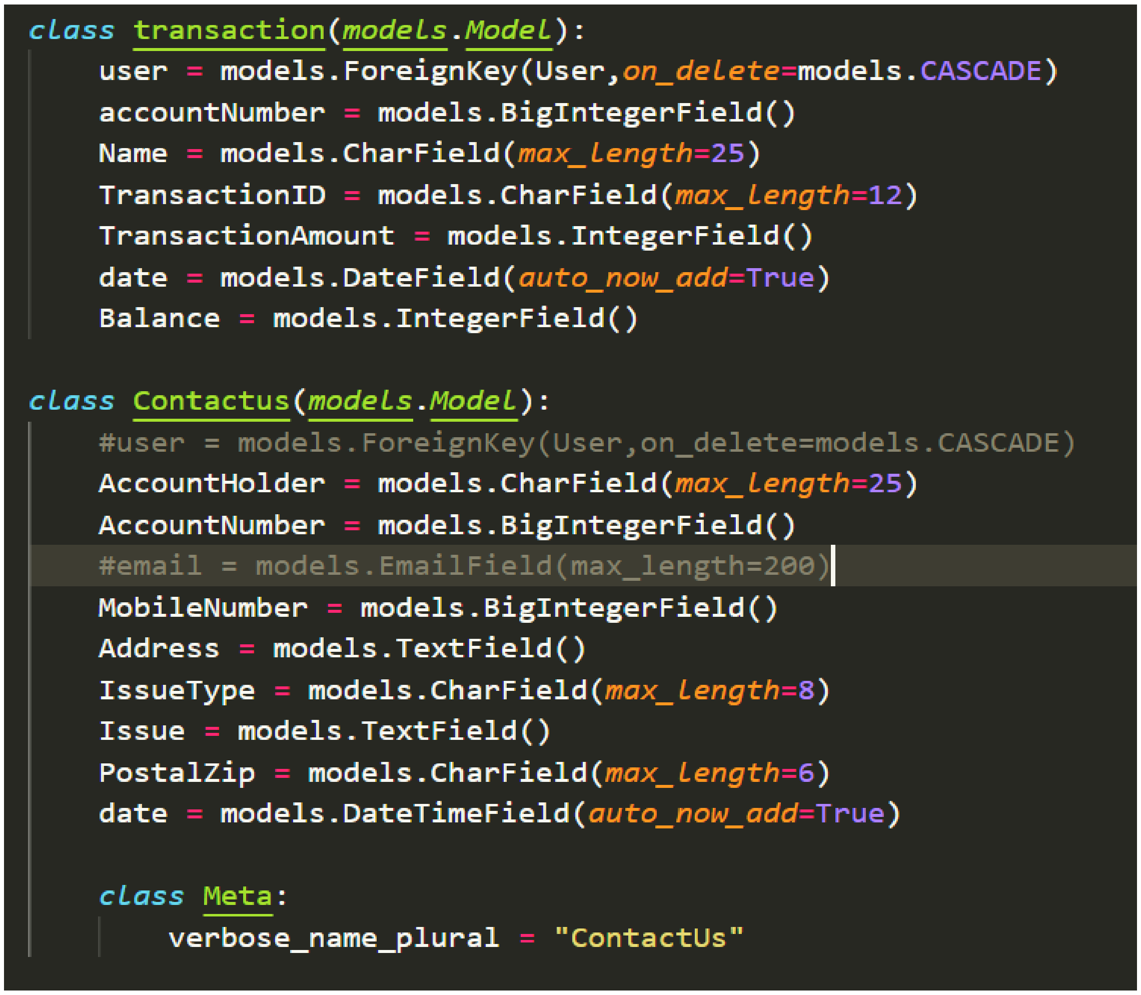


Transfer Money:

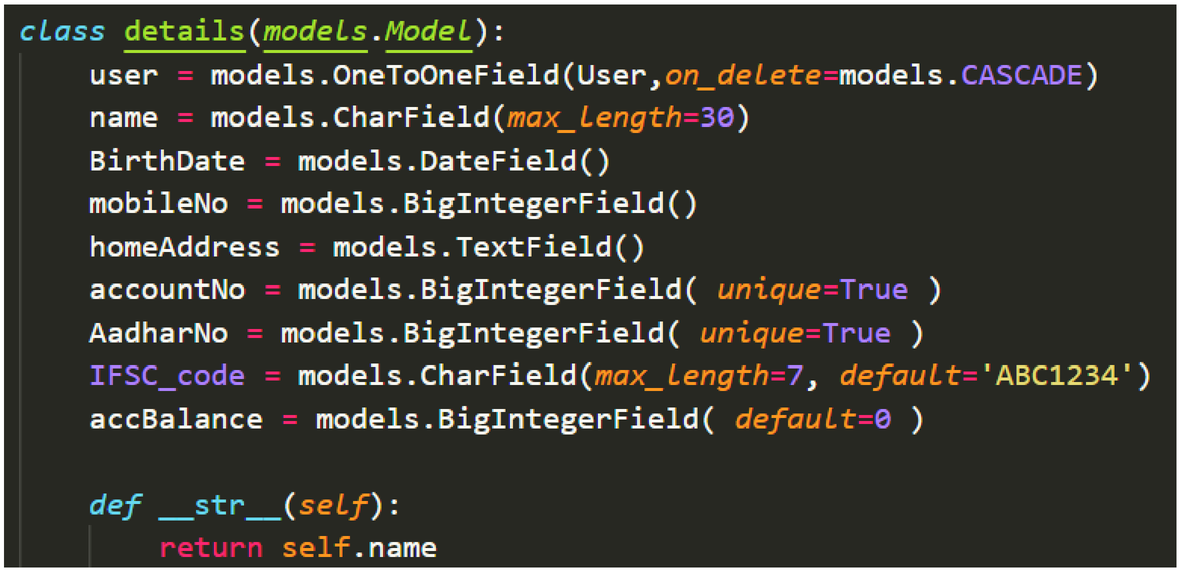


4. Implementation Details

Modals:

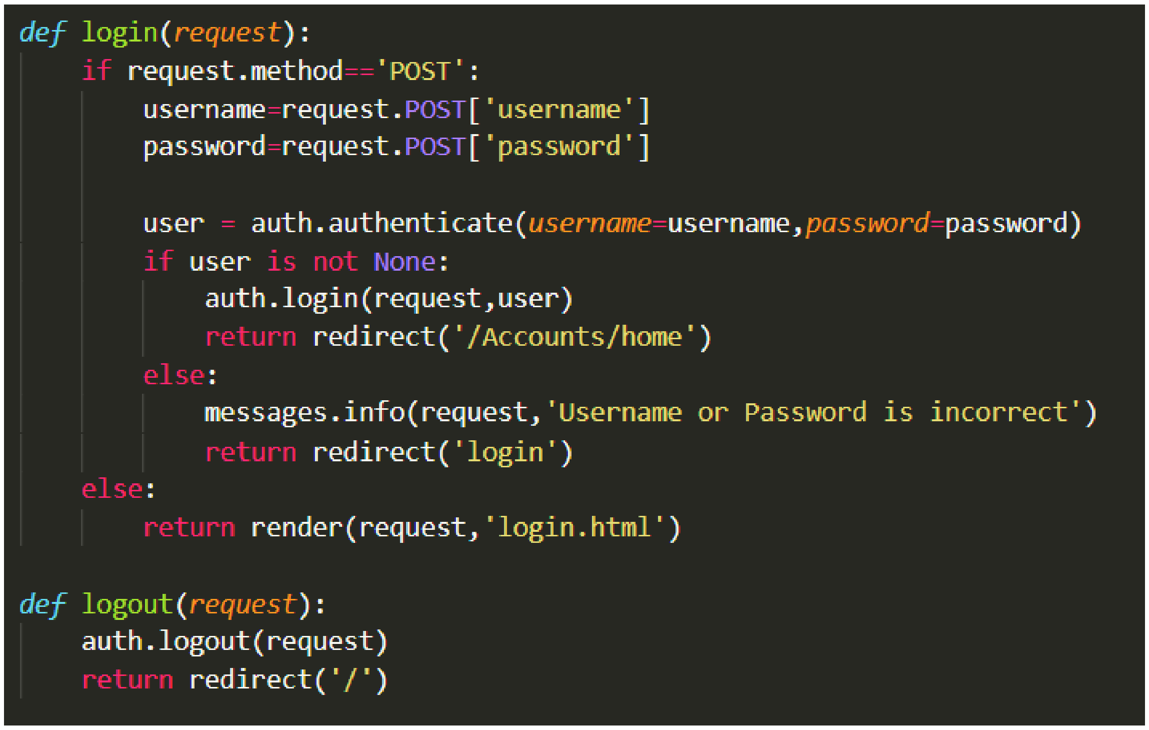


```
class transaction(models.Model):
    user = models.ForeignKey(User,on_delete=models.CASCADE)
    accountNumber = models.BigIntegerField()
    Name = models.CharField(max_length=25)
    TransactionID = models.CharField(max_length=12)
    TransactionAmount = models.IntegerField()
    date = models.DateField(auto_now_add=True)
    Balance = models.IntegerField()

class Contactus(models.Model):
    #user = models.ForeignKey(User,on_delete=models.CASCADE)
    AccountHolder = models.CharField(max_length=25)
    AccountNumber = models.BigIntegerField()
    #email = models.EmailField(max_length=200)
    MobileNumber = models.BigIntegerField()
    Address = models.TextField()
    IssueType = models.CharField(max_length=8)
    Issue = models.TextField()
    PostalZip = models.CharField(max_length=6)
    date = models.DateTimeField(auto_now_add=True)

    class Meta:
        verbose_name_plural = "ContactUs"
```

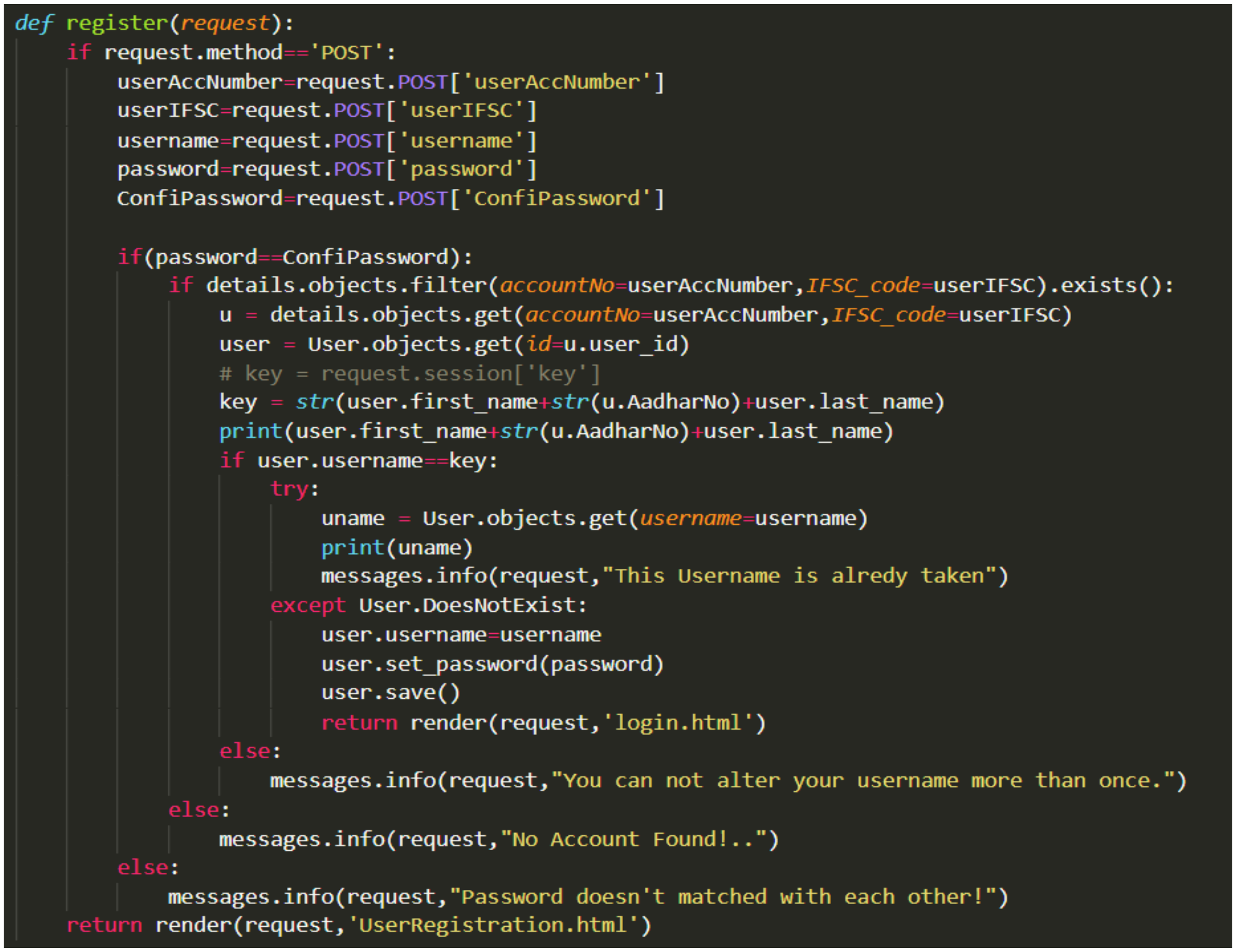


```
class details(models.Model):
    user = models.OneToOneField(User,on_delete=models.CASCADE)
    name = models.CharField(max_length=30)
    BirthDate = models.DateField()
    mobileNo = models.BigIntegerField()
    homeAddress = models.TextField()
    accountNo = models.BigIntegerField( unique=True )
    AadharNo = models.BigIntegerField( unique=True )
    IFSC_code = models.CharField(max_length=7, default='ABC1234')
    accBalance = models.BigIntegerField( default=0 )

    def __str__(self):
        return self.name
```

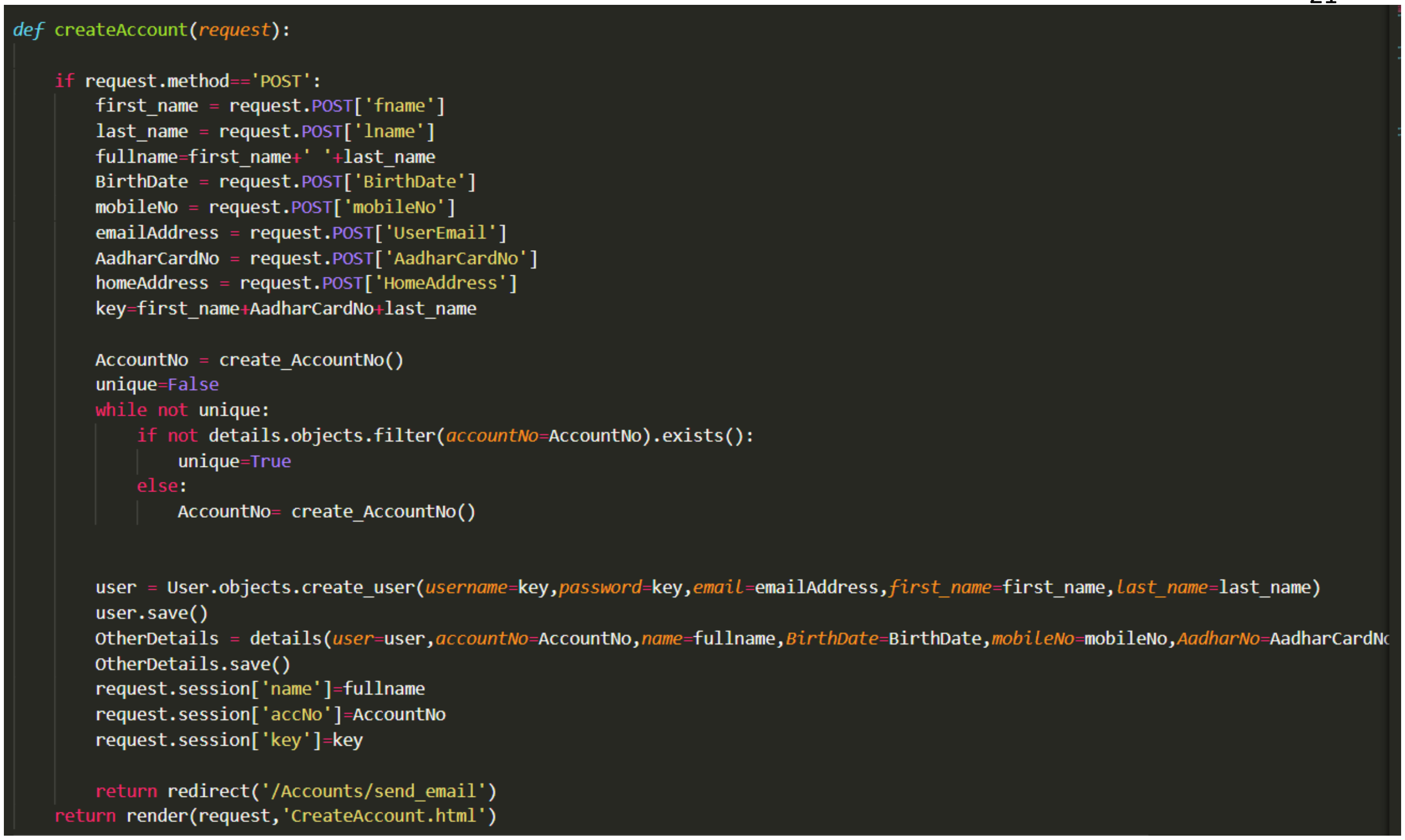
Views:



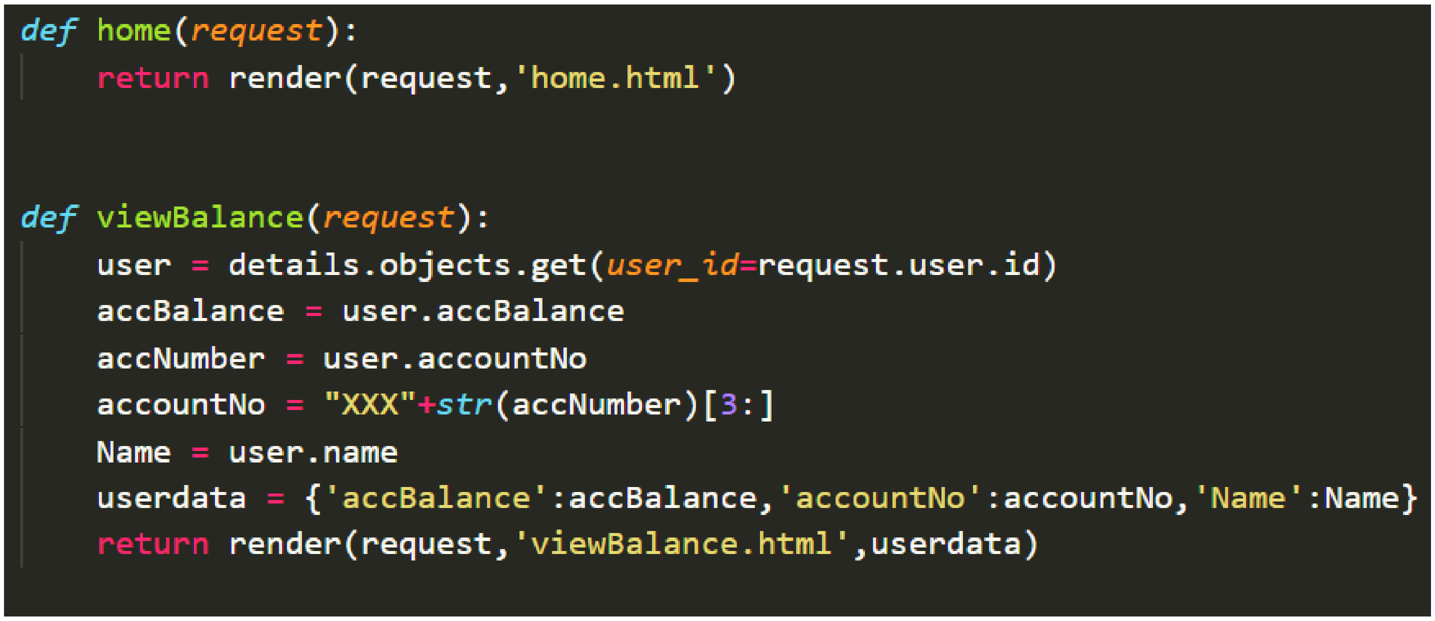
```
def login(request):
    if request.method=='POST':
        username=request.POST['username']
        password=request.POST['password']

        user = auth.authenticate(username=username,password=password)
        if user is not None:
            auth.login(request,user)
            return redirect('/Accounts/home')
        else:
            messages.info(request,'Username or Password is incorrect')
            return redirect('login')
    else:
        return render(request,'login.html')

def logout(request):
    auth.logout(request)
    return redirect('/')
```



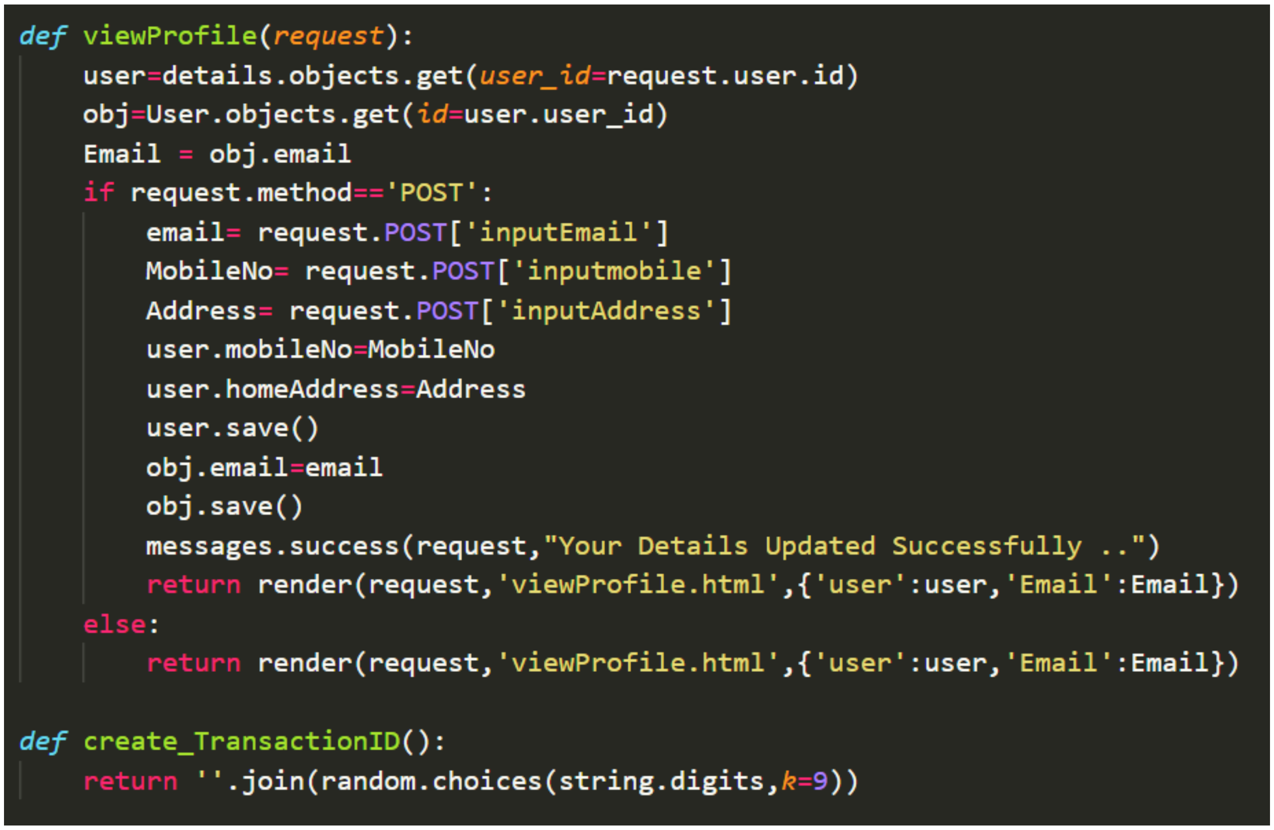
```
def register(request):
    if request.method=='POST':
        userAccNumber=request.POST['userAccNumber']
        userIFSC=request.POST['userIFSC']
        username=request.POST['username']
        password=request.POST['password']
        ConfiPassword=request.POST['ConfiPassword']

        if(password==ConfiPassword):
            if details.objects.filter(accountNo=userAccNumber,IFSC_code=userIFSC).exists():
                u = details.objects.get(accountNo=userAccNumber,IFSC_code=userIFSC)
                user = User.objects.get(id=u.user_id)
                # key = request.session['key']
                key = str(user.first_name+str(u.AadharNo)+user.last_name)
                print(user.first_name+str(u.AadharNo)+user.last_name)
                if user.username==key:
                    try:
                        uname = User.objects.get(username=username)
                        print(uname)
                        messages.info(request,"This Username is alredy taken")
                    except User.DoesNotExist:
                        user.username=username
                        user.set_password(password)
                        user.save()
                        return render(request,'login.html')
                else:
                    messages.info(request,"You can not alter your username more than once.")
            else:
                messages.info(request,"No Account Found!..")
        else:
            messages.info(request,"Password doesn't matched with each other!")
    return render(request,'UserRegistration.html')
```



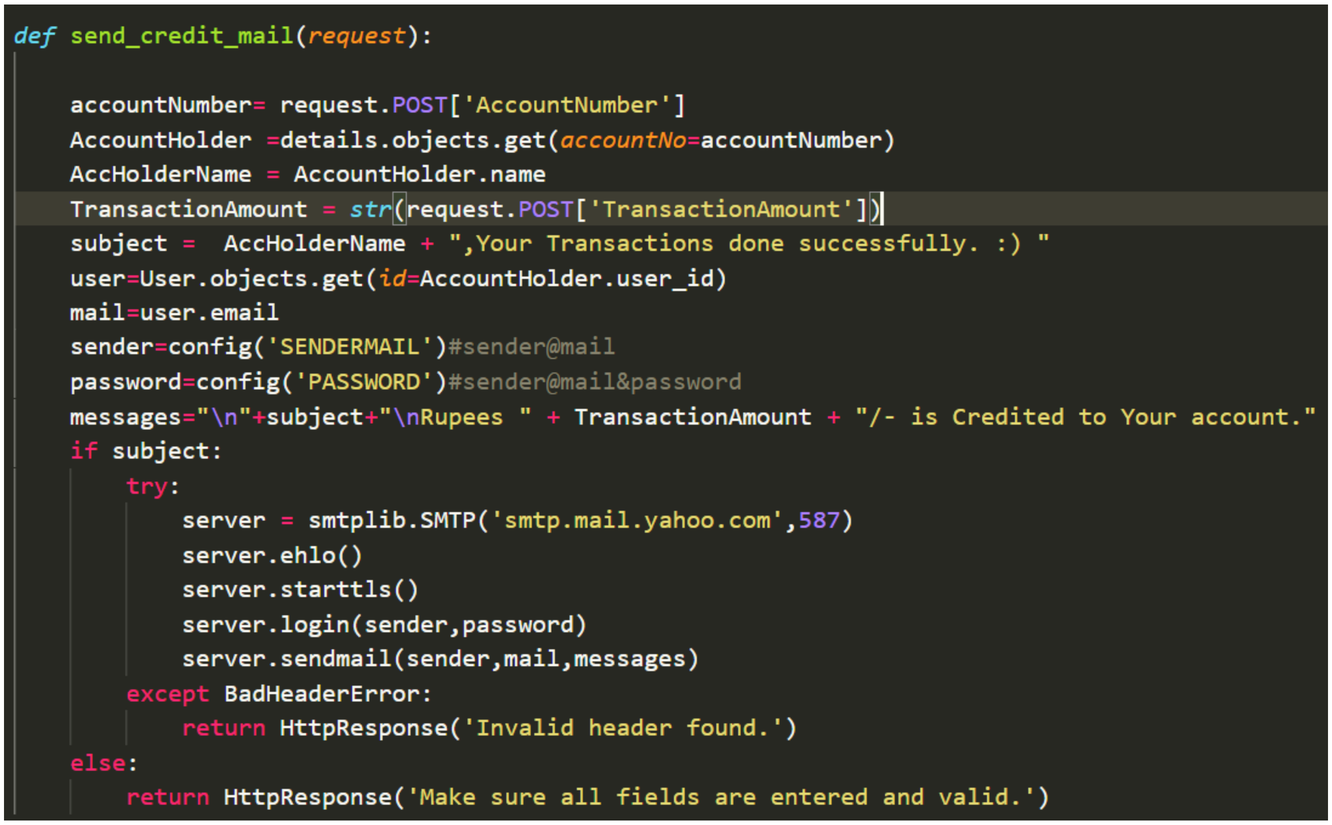
```python
def createAccount(request):

    if request.method=='POST':
        first_name = request.POST['fname']
        last_name = request.POST['lname']
        fullname=first_name+' '+last_name
        BirthDate = request.POST['BirthDate']
        mobileNo = request.POST['mobileNo']
        emailAddress = request.POST['UserEmail']
        AadharCardNo = request.POST['AadharCardNo']
        homeAddress = request.POST['HomeAddress']
        key=first_name+AadharCardNo+last_name

        AccountNo = create_AccountNo()
        unique=False
        while not unique:
            if not details.objects.filter(accountNo=AccountNo).exists():
                unique=True
            else:
                AccountNo= create_AccountNo()


        user = User.objects.create_user(username=key,password=key,email=emailAddress,first_name=first_name,last_name=last_name)
        user.save()
        OtherDetails = details(user=user,accountNo=AccountNo,name=fullname,BirthDate=BirthDate,mobileNo=mobileNo,AadharNo=AadharCardN
        OtherDetails.save()
        request.session['name']=fullname
        request.session['accNo']=AccountNo
        request.session['key']=key

        return redirect('/Accounts/send_email')
    return render(request,'CreateAccount.html')
```



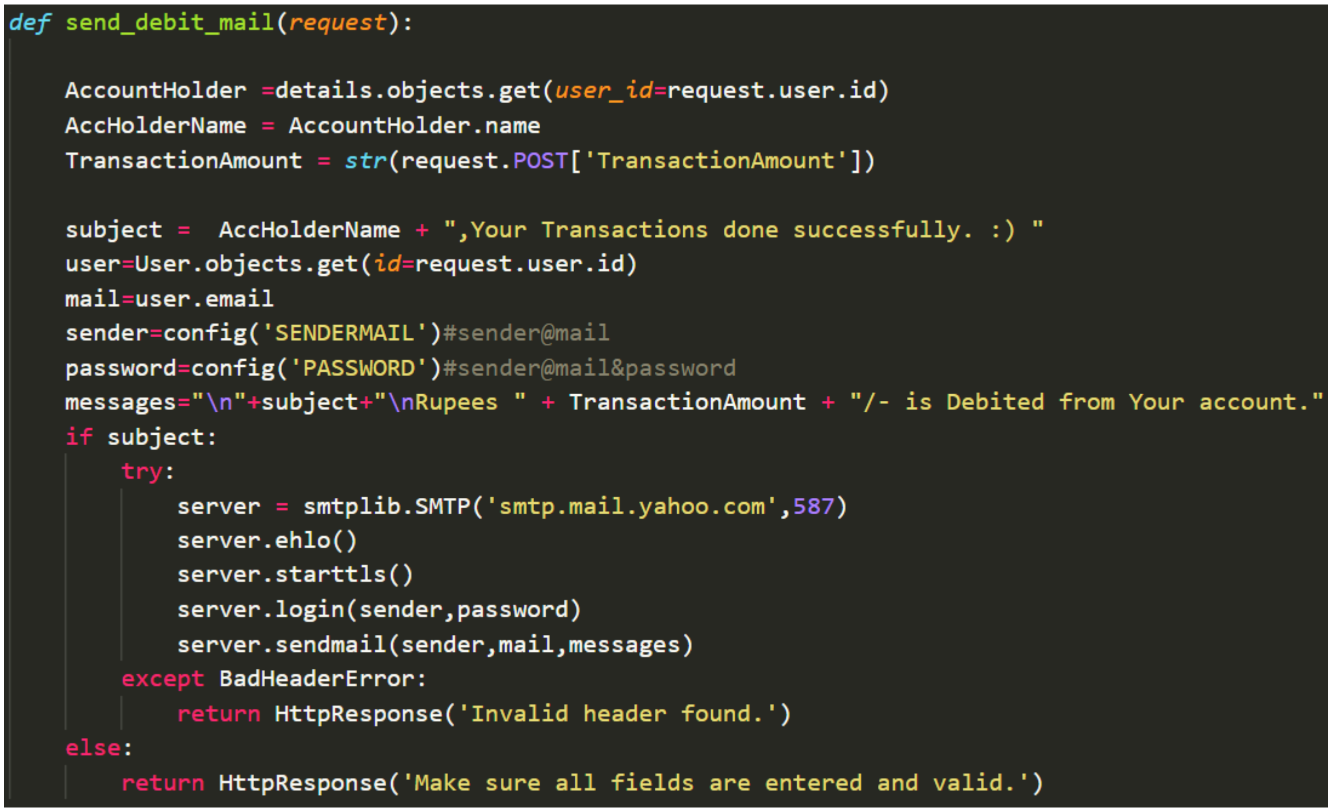
```python
def home(request):
    return render(request,'home.html')


def viewBalance(request):
    user = details.objects.get(user_id=request.user.id)
    accBalance = user.accBalance
    accNumber = user.accountNo
    accountNo = "XXX"+str(accNumber)[3:]
    Name = user.name
    userdata = {'accBalance':accBalance,'accountNo':accountNo,'Name':Name}
    return render(request,'viewBalance.html',userdata)
```



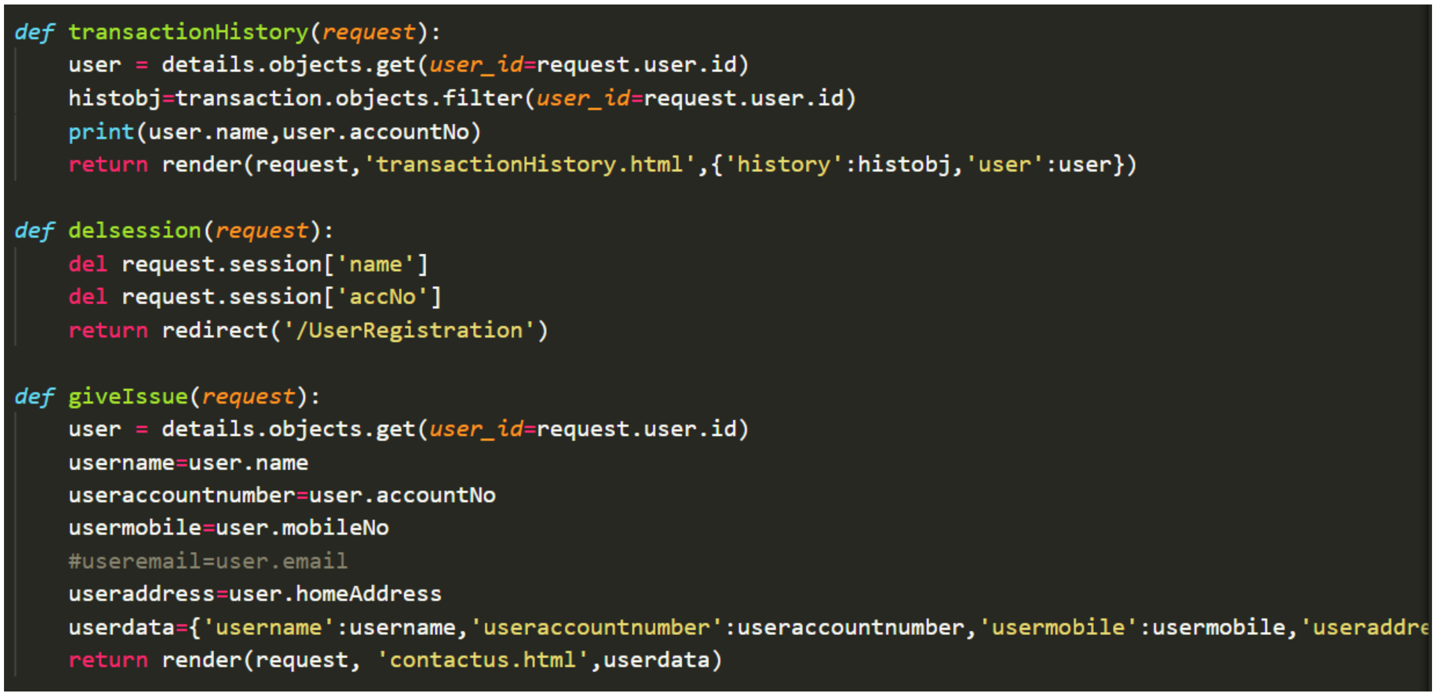
```
def viewProfile(request):
    user=details.objects.get(user_id=request.user.id)
    obj=User.objects.get(id=user.user_id)
    Email = obj.email
    if request.method=='POST':
        email= request.POST['inputEmail']
        MobileNo= request.POST['inputmobile']
        Address= request.POST['inputAddress']
        user.mobileNo=MobileNo
        user.homeAddress=Address
        user.save()
        obj.email=email
        obj.save()
        messages.success(request,"Your Details Updated Successfully ..")
        return render(request,'viewProfile.html',{'user':user,'Email':Email})
    else:
        return render(request,'viewProfile.html',{'user':user,'Email':Email})

def create_TransactionID():
    return ''.join(random.choices(string.digits,k=9))
```



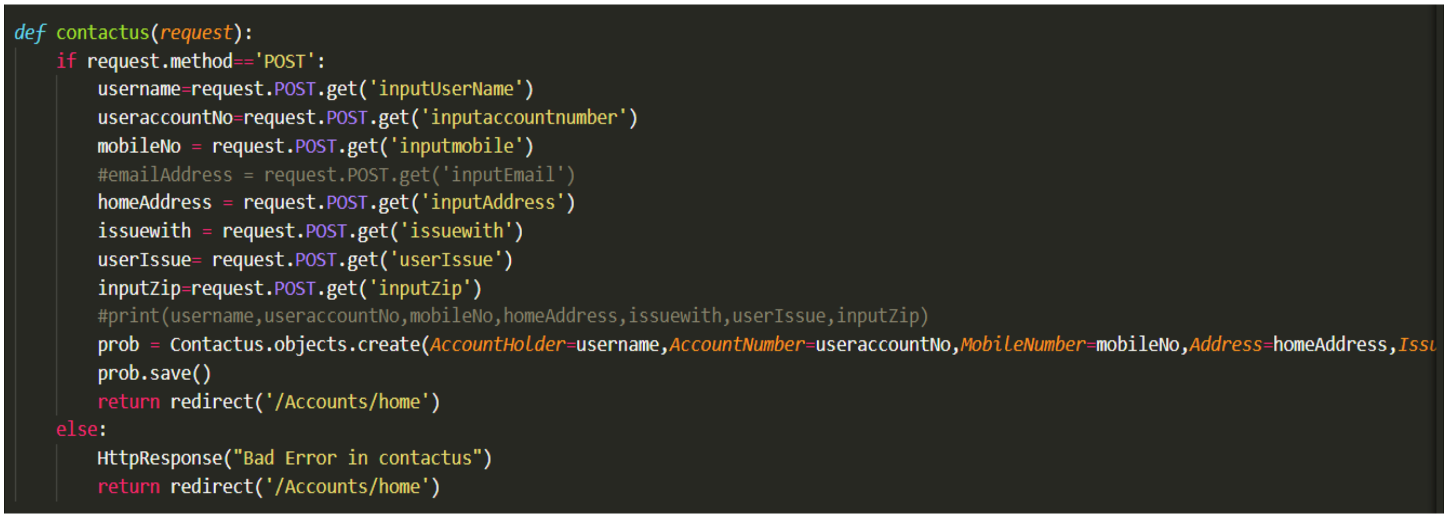
```
def send_credit_mail(request):

    accountNumber= request.POST['AccountNumber']
    AccountHolder =details.objects.get(accountNo=accountNumber)
    AccHolderName = AccountHolder.name
    TransactionAmount = str(request.POST['TransactionAmount'])
    subject =  AccHolderName + ",Your Transactions done successfully. :) "
    user=User.objects.get(id=AccountHolder.user_id)
    mail=user.email
    sender=config('SENDERMAIL')#sender@mail
    password=config('PASSWORD')#sender@mail&password
    messages="\n"+subject+"\nRupees " + TransactionAmount + "/- is Credited to Your account."
    if subject:
        try:
            server = smtplib.SMTP('smtp.mail.yahoo.com',587)
            server.ehlo()
            server.starttls()
            server.login(sender,password)
            server.sendmail(sender,mail,messages)
        except BadHeaderError:
            return HttpResponse('Invalid header found.')
    else:
        return HttpResponse('Make sure all fields are entered and valid.')
```



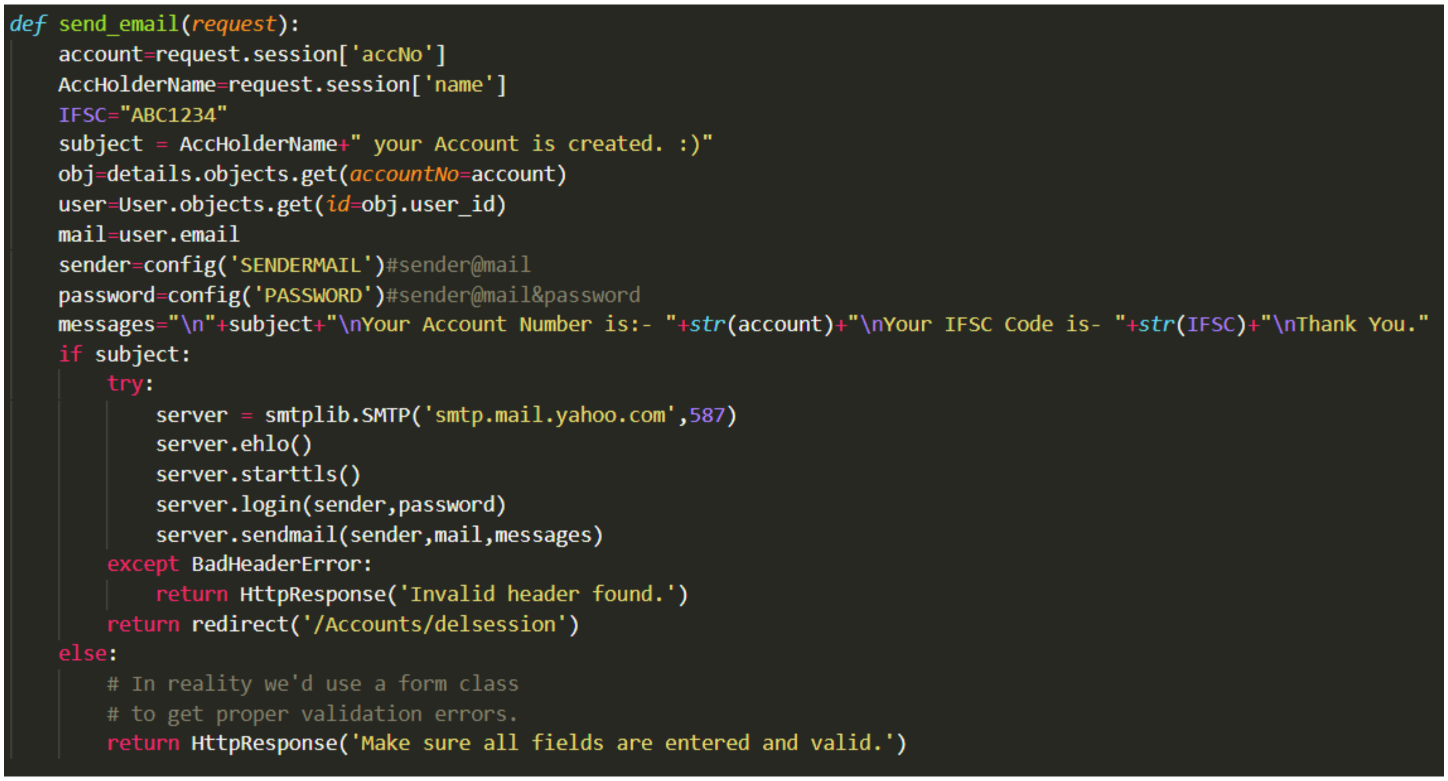
```
def send_debit_mail(request):

    AccountHolder =details.objects.get(user_id=request.user.id)
    AccHolderName = AccountHolder.name
    TransactionAmount = str(request.POST['TransactionAmount'])

    subject =  AccHolderName + ",Your Transactions done successfully. :) "
    user=User.objects.get(id=request.user.id)
    mail=user.email
    sender=config('SENDERMAIL')#sender@mail
    password=config('PASSWORD')#sender@mail&password
    messages="\n"+subject+"\nRupees " + TransactionAmount + "/- is Debited from Your account."
    if subject:
        try:
            server = smtplib.SMTP('smtp.mail.yahoo.com',587)
            server.ehlo()
            server.starttls()
            server.login(sender,password)
            server.sendmail(sender,mail,messages)
        except BadHeaderError:
            return HttpResponse('Invalid header found.')
    else:
        return HttpResponse('Make sure all fields are entered and valid.')
```



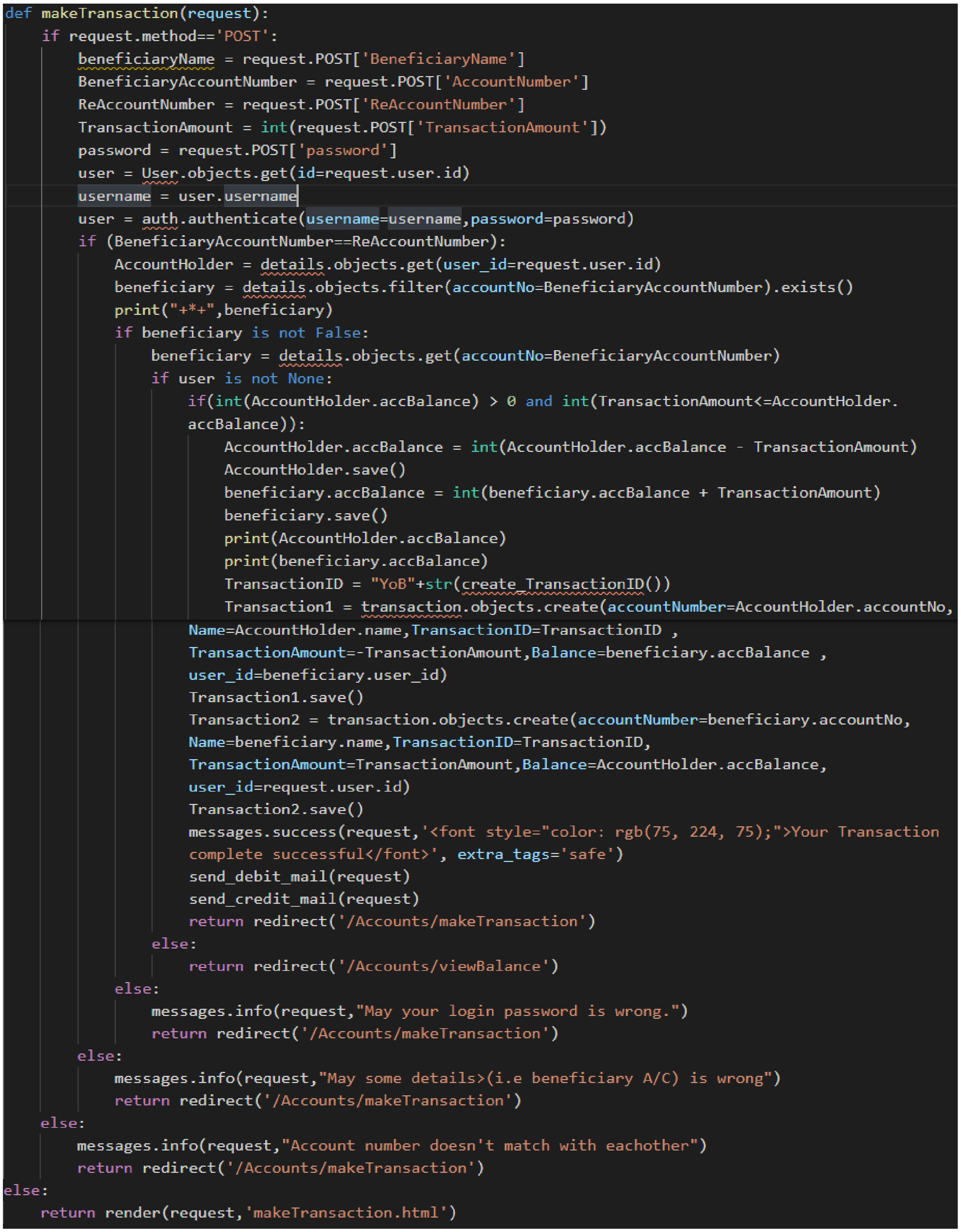
```
def transactionHistory(request):
    user = details.objects.get(user_id=request.user.id)
    histobj=transaction.objects.filter(user_id=request.user.id)
    print(user.name,user.accountNo)
    return render(request,'transactionHistory.html',{'history':histobj,'user':user})

def delsession(request):
    del request.session['name']
    del request.session['accNo']
    return redirect('/UserRegistration')

def giveIssue(request):
    user = details.objects.get(user_id=request.user.id)
    username=user.name
    useraccountnumber=user.accountNo
    usermobile=user.mobileNo
    #useremail=user.email
    useraddress=user.homeAddress
    userdata={'username':username,'useraccountnumber':useraccountnumber,'usermobile':usermobile,'useraddre
    return render(request, 'contactus.html',userdata)
```



```python
def contactus(request):
    if request.method=='POST':
        username=request.POST.get('inputUserName')
        useraccountNo=request.POST.get('inputaccountnumber')
        mobileNo = request.POST.get('inputmobile')
        #emailAddress = request.POST.get('inputEmail')
        homeAddress = request.POST.get('inputAddress')
        issuewith = request.POST.get('issuewith')
        userIssue= request.POST.get('userIssue')
        inputZip=request.POST.get('inputZip')
        #print(username,useraccountNo,mobileNo,homeAddress,issuewith,userIssue,inputZip)
        prob = Contactus.objects.create(AccountHolder=username,AccountNumber=useraccountNo,MobileNumber=mobileNo,Address=homeAddress,Issu
        prob.save()
        return redirect('/Accounts/home')
    else:
        HttpResponse("Bad Error in contactus")
        return redirect('/Accounts/home')
```



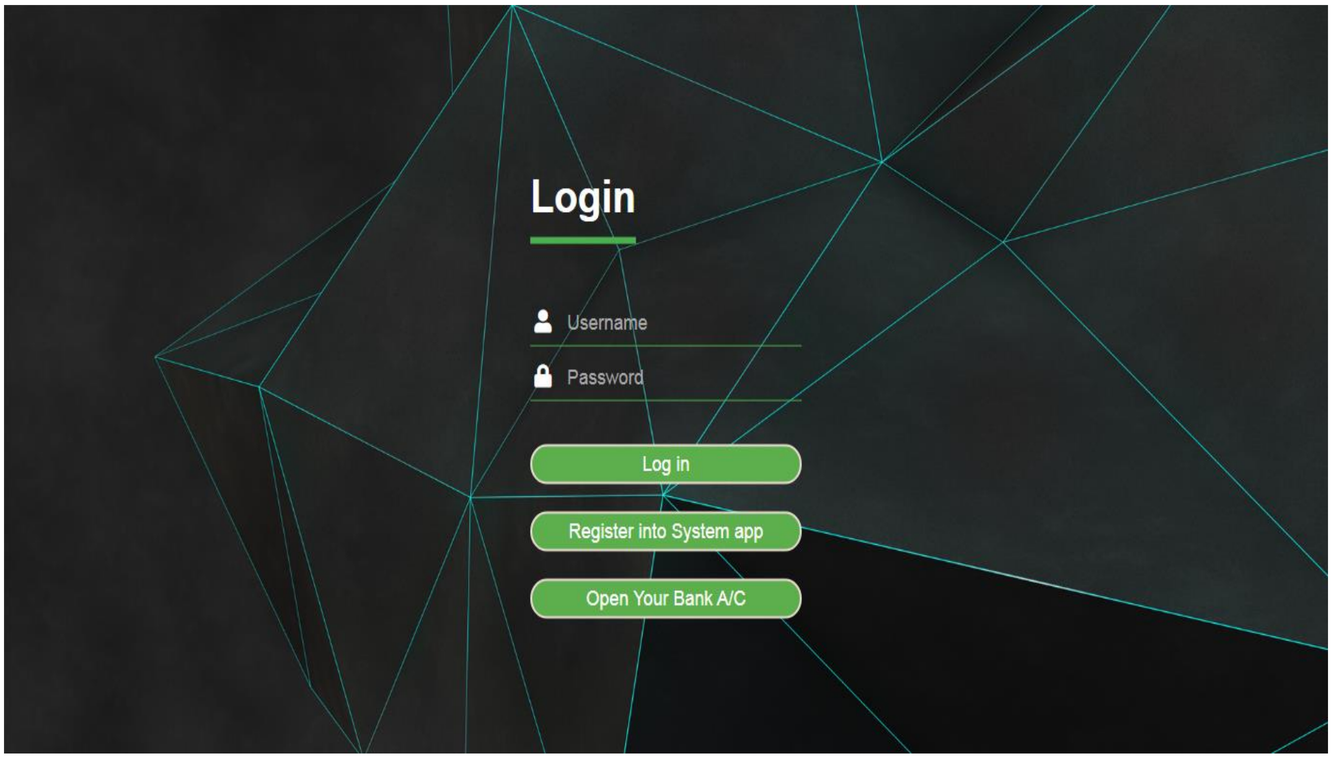
```python
def send_email(request):
    account=request.session['accNo']
    AccHolderName=request.session['name']
    IFSC="ABC1234"
    subject = AccHolderName+" your Account is created. :)"
    obj=details.objects.get(accountNo=account)
    user=User.objects.get(id=obj.user_id)
    mail=user.email
    sender=config('SENDERMAIL')#sender@mail
    password=config('PASSWORD')#sender@mail&password
    messages="\n"+subject+"\nYour Account Number is:- "+str(account)+"\nYour IFSC Code is- "+str(IFSC)+"\nThank You."
    if subject:
        try:
            server = smtplib.SMTP('smtp.mail.yahoo.com',587)
            server.ehlo()
            server.starttls()
            server.login(sender,password)
            server.sendmail(sender,mail,messages)
        except BadHeaderError:
            return HttpResponse('Invalid header found.')
        return redirect('/Accounts/delsession')
    else:
        # In reality we'd use a form class
        # to get proper validation errors.
        return HttpResponse('Make sure all fields are entered and valid.')
```



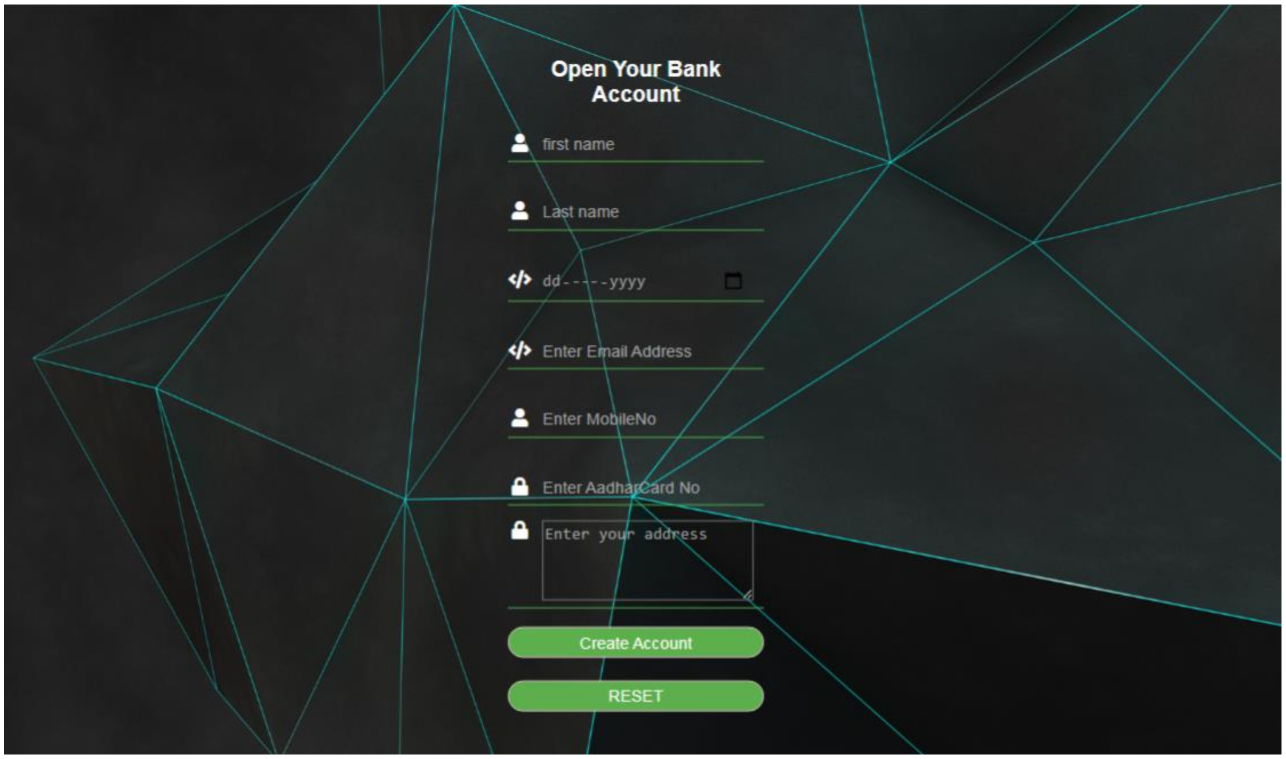
```python
def makeTransaction(request):
    if request.method=='POST':
        beneficiaryName = request.POST['BeneficiaryName']
        BeneficiaryAccountNumber = request.POST['AccountNumber']
        ReAccountNumber = request.POST['ReAccountNumber']
        TransactionAmount = int(request.POST['TransactionAmount'])
        password = request.POST['password']
        user = User.objects.get(id=request.user.id)
        username = user.username
        user = auth.authenticate(username=username,password=password)
        if (BeneficiaryAccountNumber==ReAccountNumber):
            AccountHolder = details.objects.get(user_id=request.user.id)
            beneficiary = details.objects.filter(accountNo=BeneficiaryAccountNumber).exists()
            print("+*+",beneficiary)
            if beneficiary is not False:
                beneficiary = details.objects.get(accountNo=BeneficiaryAccountNumber)
                if user is not None:
                    if(int(AccountHolder.accBalance) > 0 and int(TransactionAmount<=AccountHolder.
                    accBalance)):
                        AccountHolder.accBalance = int(AccountHolder.accBalance - TransactionAmount)
                        AccountHolder.save()
                        beneficiary.accBalance = int(beneficiary.accBalance + TransactionAmount)
                        beneficiary.save()
                        print(AccountHolder.accBalance)
                        print(beneficiary.accBalance)
                        TransactionID = "YoB"+str(create_TransactionID())
                        Transaction1 = transaction.objects.create(accountNumber=AccountHolder.accountNo,
                    Name=AccountHolder.name,TransactionID=TransactionID ,
                    TransactionAmount=-TransactionAmount,Balance=beneficiary.accBalance ,
                    user_id=beneficiary.user_id)
                    Transaction1.save()
                    Transaction2 = transaction.objects.create(accountNumber=beneficiary.accountNo,
                    Name=beneficiary.name,TransactionID=TransactionID,
                    TransactionAmount=TransactionAmount,Balance=AccountHolder.accBalance,
                    user_id=request.user.id)
                    Transaction2.save()
                    messages.success(request,'<font style="color: rgb(75, 224, 75);">Your Transaction
                    complete successful</font>', extra_tags='safe')
                    send_debit_mail(request)
                    send_credit_mail(request)
                    return redirect('/Accounts/makeTransaction')
                else:
                    return redirect('/Accounts/viewBalance')
            else:
                messages.info(request,"May your login password is wrong.")
                return redirect('/Accounts/makeTransaction')
        else:
            messages.info(request,"May some details>(i.e beneficiary A/C) is wrong")
            return redirect('/Accounts/makeTransaction')
    else:
        messages.info(request,"Account number doesn't match with eachother")
        return redirect('/Accounts/makeTransaction')
else:
    return render(request,'makeTransaction.html')
```

5. Workflow / Layouts

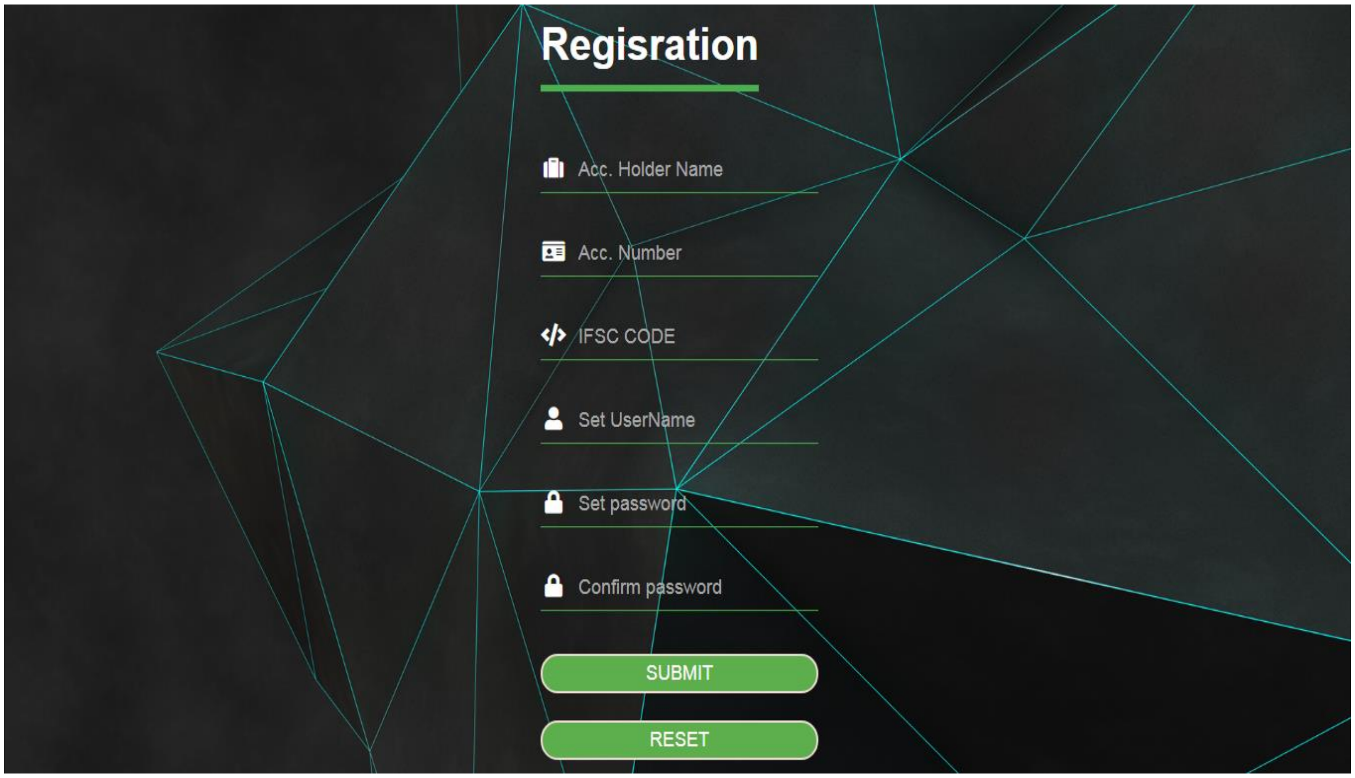
1. Login



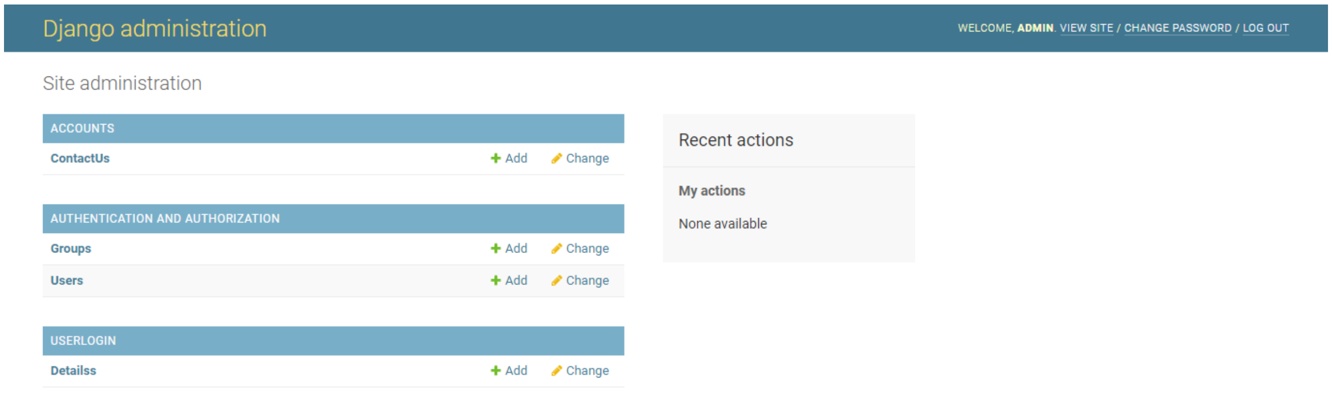
2. Create Account



3. User Registration:



4. Admin Panel



Django administration

WELCOME, **ADMIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

ACCOUNTS		
ContactUs	+ Add	Change

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

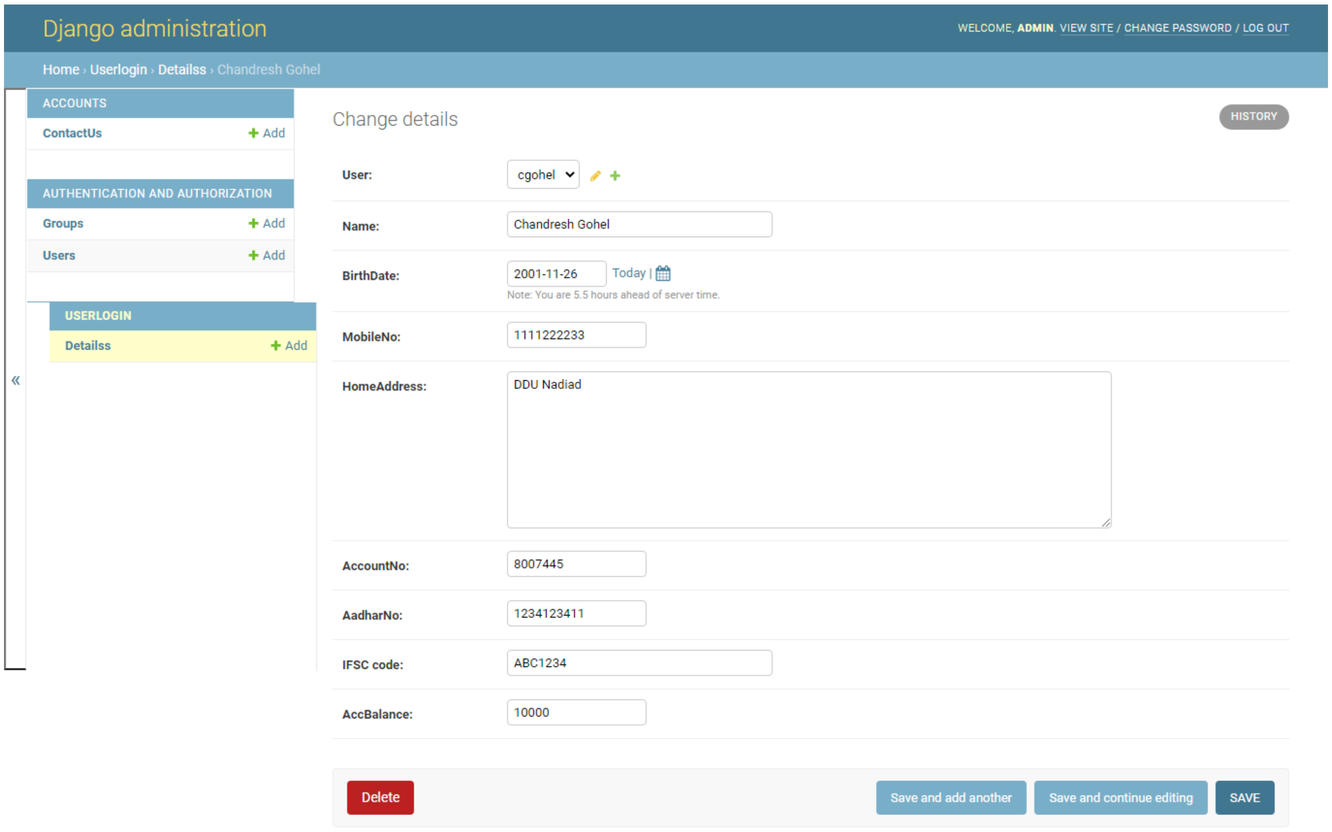
USERLOGIN		
Detailss	+ Add	Change

Recent actions

My actions

None available

5. Amin User Profile



Django administration

WELCOME, **ADMIN**. VIEW SITE / CHANGE PASSWORD / LOG OUT

Home › Userlogin › Detailss › Chandresh Gohel

ACCOUNTS	
ContactUs	+ Add

AUTHENTICATION AND AUTHORIZATION	
Groups	+ Add
Users	+ Add

USERLOGIN	
Detailss	+ Add

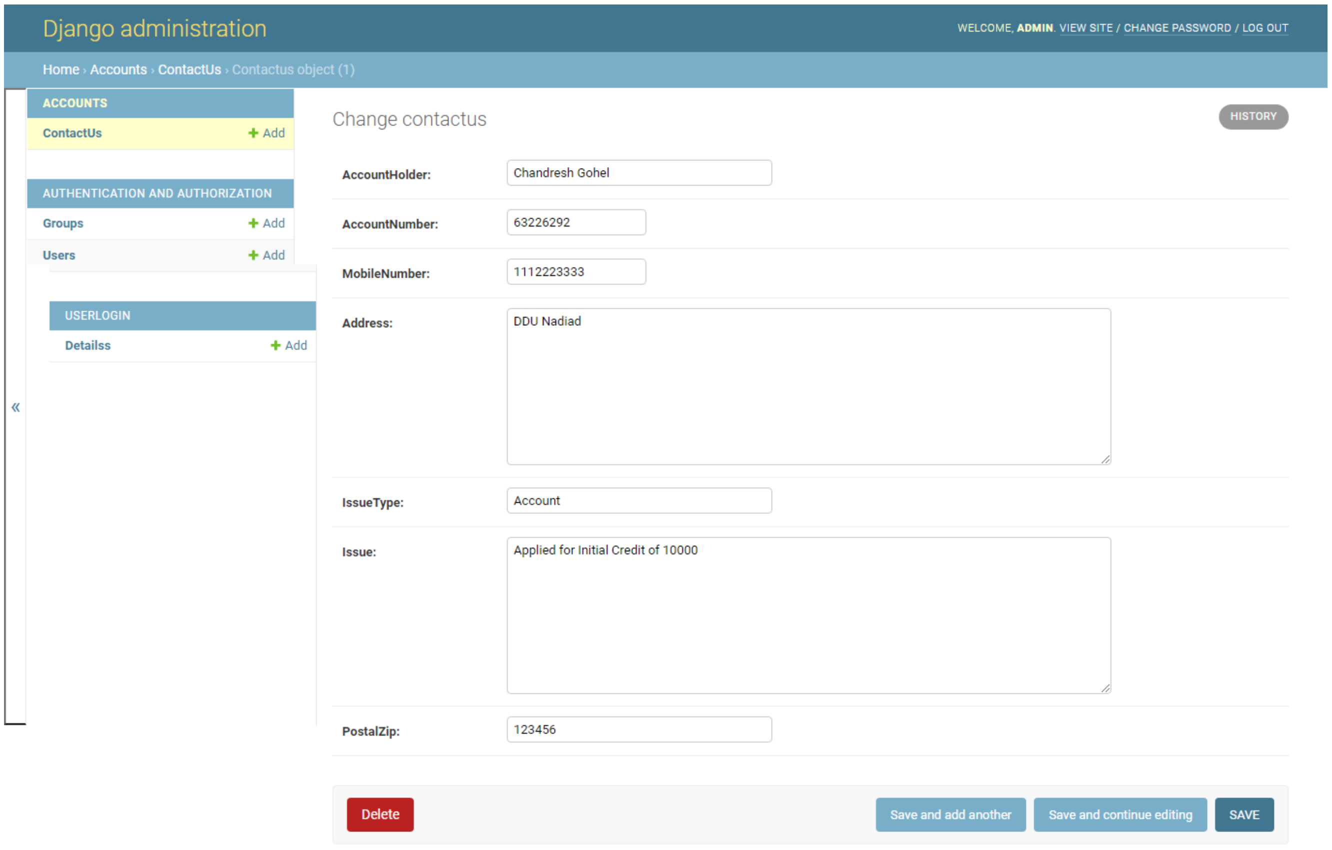
Change details

HISTORY

User:	cgohel
Name:	Chandresh Gohel
BirthDate:	2001-11-26 Today | Note: You are 5.5 hours ahead of server time.
MobileNo:	1111222233
HomeAddress:	DDU Nadiad
AccountNo:	8007445
AadharNo:	1234123411
IFSC code:	ABC1234
AccBalance:	10000

Delete | Save and add another | Save and continue editing | SAVE

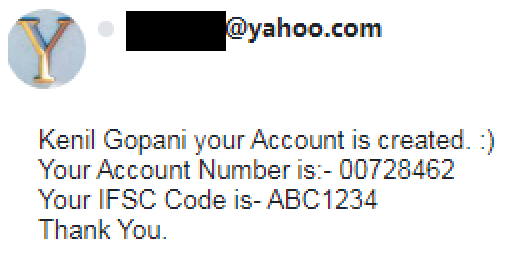
6. Admin Contact Us


Django administration
WELCOME, ADMIN. VIEW SITE / CHANGE PASSWORD / LOG OUT
Home › Accounts › ContactUs › Contactus object (1)
ACCOUNTS
ContactUs + Add
AUTHENTICATION AND AUTHORIZATION
Groups + Add
Users + Add
USERLOGIN
Detailss + Add
«
HISTORY
Delete
Save and add another
Save and continue editing
SAVE

Change contactus

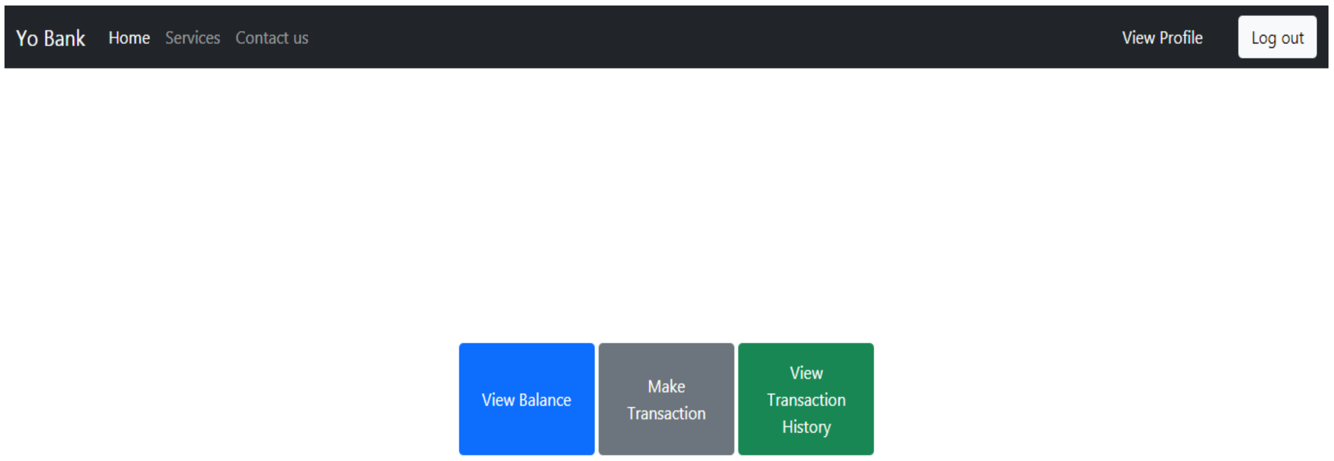
AccountHolder:	Chandresh Gohel
AccountNumber:	63226292
MobileNumber:	1112223333
Address:	DDU Nadiad
IssueType:	Account
Issue:	Applied for Initial Credit of 10000
PostalZip:	123456

7. Account Created Mail

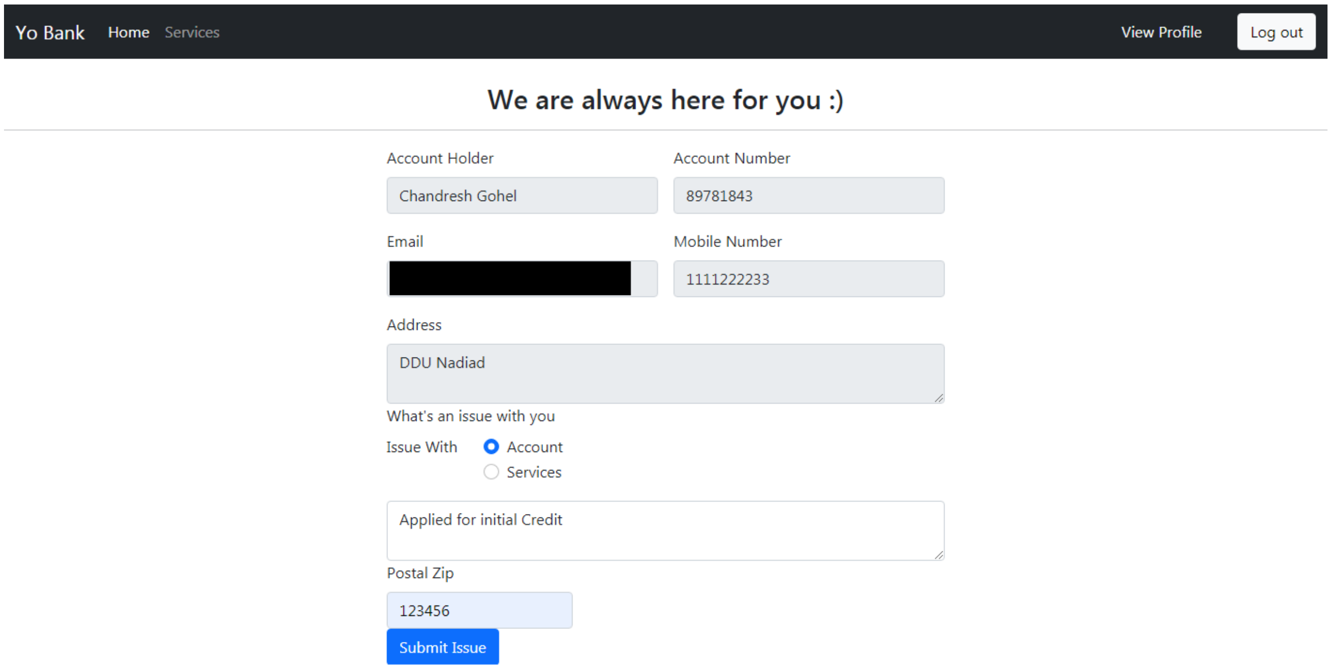

[redacted]@yahoo.com

Kenil Gopani your Account is created. :)
Your Account Number is:- 00728462
Your IFSC Code is- ABC1234
Thank You.

8. Home Page



9. Contact Us (User)



We are always here for you :)

Account Holder	Account Number
Chandresh Gohel	89781843
Email	Mobile Number
[redacted]	1111222233

Address: DDU Nadiad

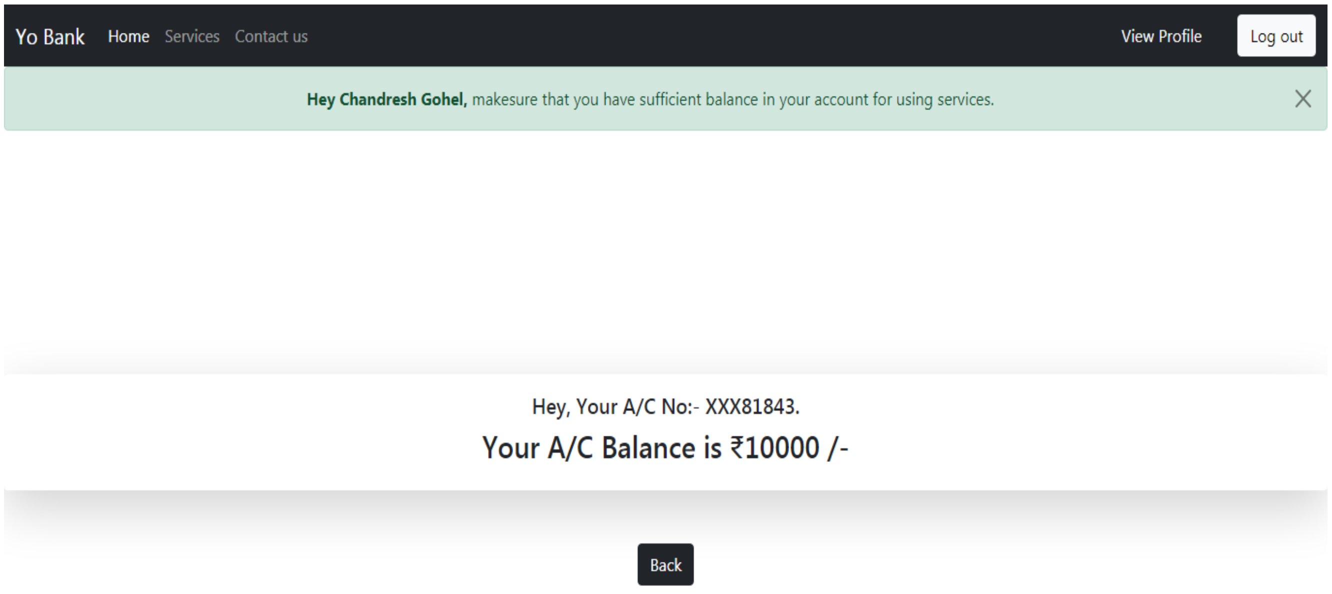
What's an issue with you

Issue With [✓ Account] Services

Applied for initial Credit

Postal Zip: 123456

10. View Balance

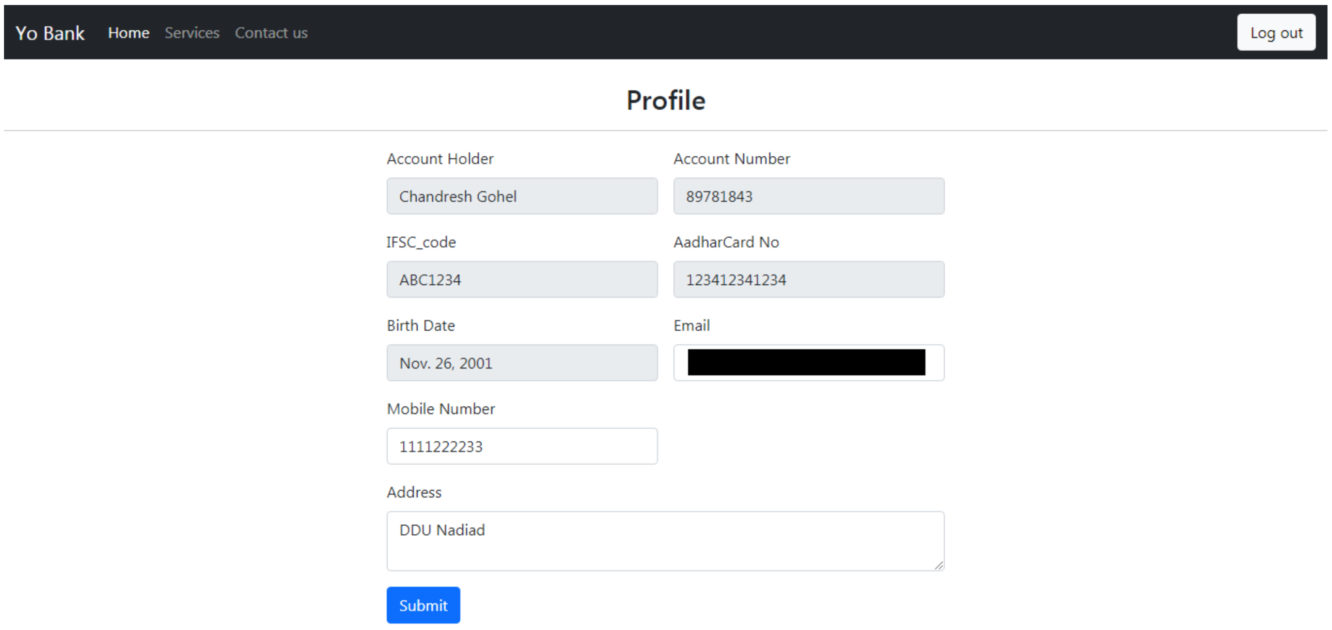


Hey Chandresh Gohel, makesure that you have sufficient balance in your account for using services.

Hey, Your A/C No:- XXX81843.

Your A/C Balance is ₹10000 /-

11. View/Update Profile



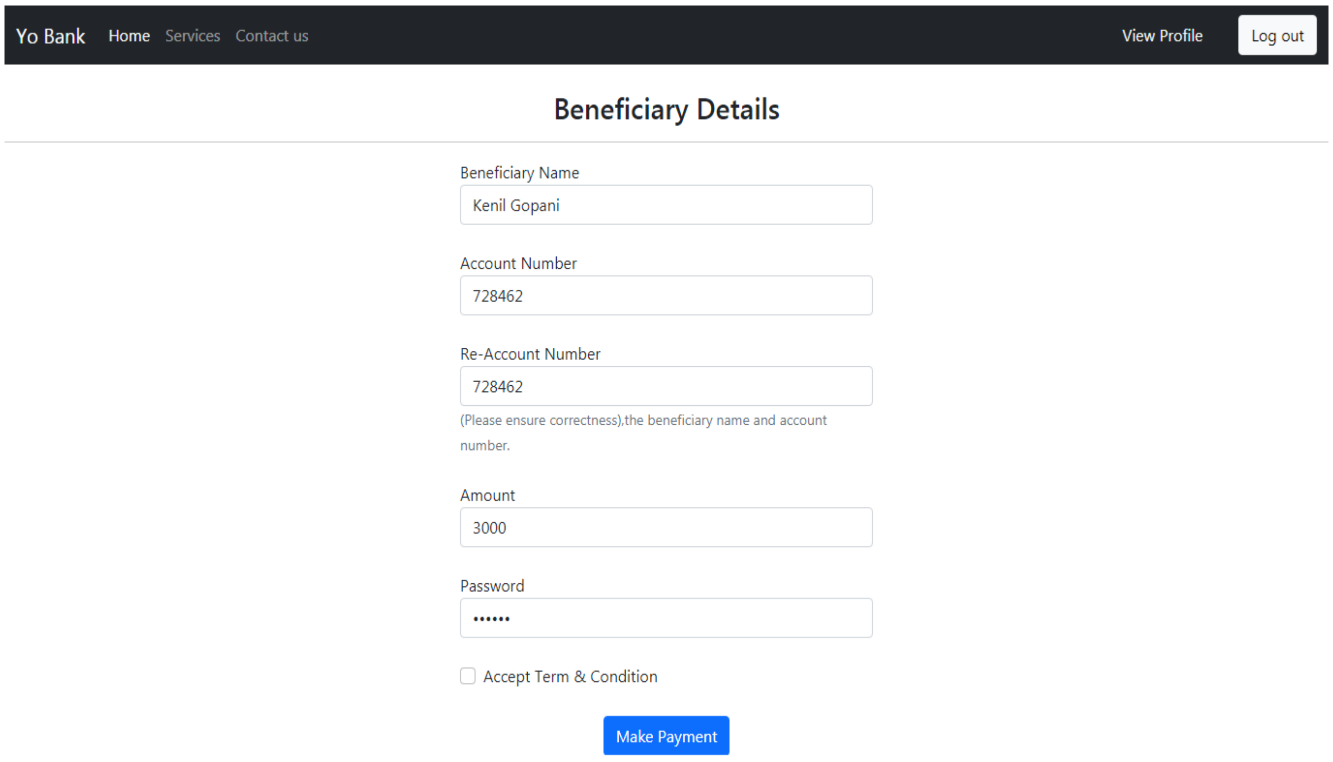
Profile

Account Holder	Account Number
Chandresh Gohel	89781843
IFSC_code	AadharCard No
ABC1234	123412341234
Birth Date	Email
Nov. 26, 2001	[redacted]
Mobile Number	
1111222233	

Address

DDU Nadiad

12. Transfer Money



Beneficiary Details

Beneficiary Name
Kenil Gopani

Account Number
728462

Re-Account Number
728462

(Please ensure correctness),the beneficiary name and account number.

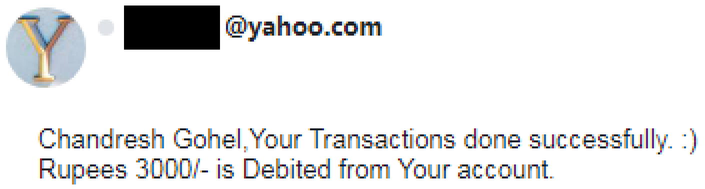
Amount
3000

Password
••••••

Accept Term & Condition

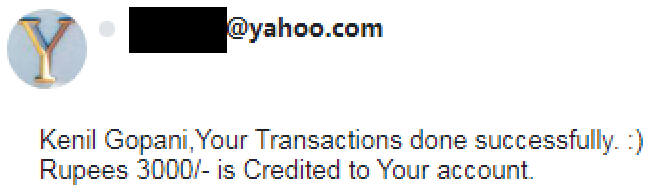
13. Transaction Information:

1: From Account:


[redacted]@yahoo.com

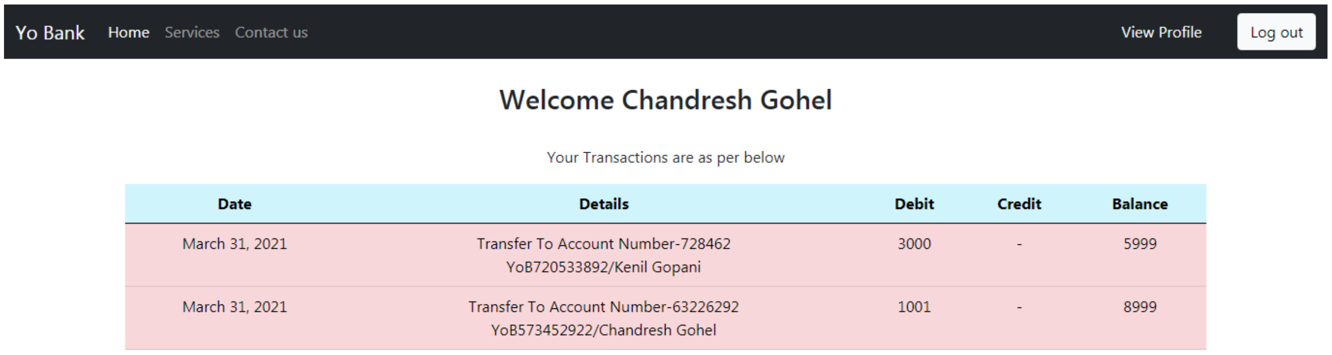
Chandresh Gohel,Your Transactions done successfully. :)
Rupees 3000/- is Debited from Your account.

2: To Account


[redacted]@yahoo.com

Kenil Gopani,Your Transactions done successfully. :)
Rupees 3000/- is Credited to Your account.

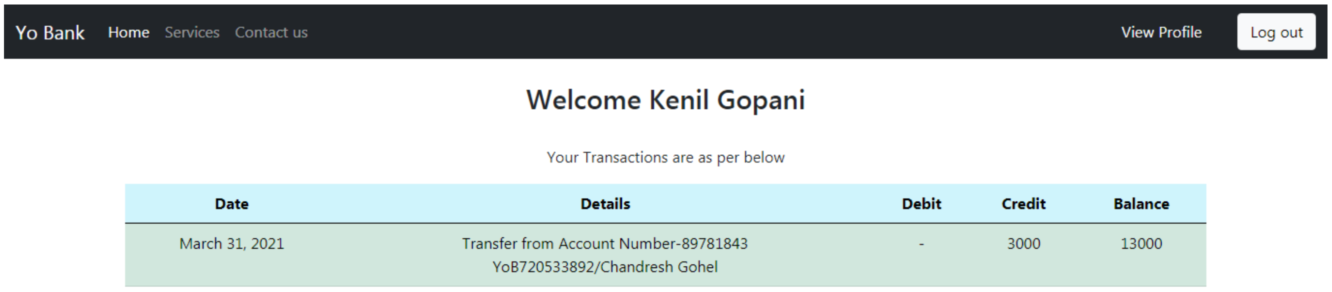
14. Transaction History:



Welcome Chandresh Gohel

Your Transactions are as per below

Date	Details	Debit	Credit	Balance
March 31, 2021	Transfer To Account Number-728462 YoB720533892/Kenil Gopani	3000	-	5999
March 31, 2021	Transfer To Account Number-63226292 YoB573452922/Chandresh Gohel	1001	-	8999



Welcome Kenil Gopani

Your Transactions are as per below

Date	Details	Debit	Credit	Balance
March 31, 2021	Transfer from Account Number-89781843 YoB720533892/Chandresh Gohel	-	3000	13000

6. Conclusion

All the functionalities are implemented after understanding all models and diagrams of system. By the use of system all users of bank can interact with banking services and bank admins, increased user ease.

Functionalities are successfully implemented are as below mentioned.

- Login
- User Registration
- Create Account
- View Balance
- Make Transaction
- Inform User transaction
- Transaction History
- View Profile
- Update Profile
- Contact Admin

After the implementation all functionalities were successfully tested, and working properly.

7. Limitation and future extension

Limitation:

- In this system we can make transaction among only who belongs to same bank. Not with other banks, as made for Local System.
- Current system users can’t delete their account directly from user panel.
- Current system doesn’t define limitation of transaction amount.

Future Extension:

- User interface will be improved to provide better interaction with system.
- In further version delete account functionality.
- Services (fixed deposit, issue debit/credit card, issue cheque book) will be added.
- At time of transaction OTP functionality will be added.
- User will be able to print his/her transactions.

8. Bibliography

Referred links for project:

- www.docs.djangoproject.com
- www.stackoverflow.com
- www.askpython.com
- www.geeksforgeeks.com
- www.github.com