|  |
| --- |
| ICIN Bank  (Source Code) |

|  |  |
| --- | --- |
| Author | Aman Kumar |
| Purpose | Project Source Code |
| Date | 22/07/2022 |
| Version | 1.0 |

Version History:

**Project Code:**

**Front End:**

**User Portal:**

**Index.html:**

<!doctype html>

<html lang="en" class="full-height">

<head>

  <meta charset="utf-8">

  <title>DreamApp</title>

  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">

  <link rel="icon" type="image/x-icon" href="favicon.ico">

  <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.16.0/umd/popper.min.js"></script>

</head>

<body>

  <app-root></app-root>

</body>

</html>

**Main.ts**

import { enableProdMode } from '@angular/core';

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';

import { environment } from './environments/environment';

if (environment.production) {

  enableProdMode();

}

platformBrowserDynamic().bootstrapModule(AppModule)

  .catch(err => console.error(err));

**Test.ts**

// This file is required by karma.conf.js and loads recursively all the .spec and framework files

import 'zone.js/dist/zone-testing';

import { getTestBed } from '@angular/core/testing';

import {

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting

} from '@angular/platform-browser-dynamic/testing';

declare const require: {

  context(path: string, deep?: boolean, filter?: RegExp): {

    keys(): string[];

    <T>(id: string): T;

  };

};

// First, initialize the Angular testing environment.

getTestBed().initTestEnvironment(

  BrowserDynamicTestingModule,

  platformBrowserDynamicTesting()

);

// Then we find all the tests.

const context = require.context('./', true, /\.spec\.ts$/);

// And load the modules.

context.keys().map(context);

**login.service.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class LoginService {

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

  constructor(private http: HttpClient) { }

  loginUser(userName: string, password: string) {

    var body = {

      username: userName,

      password: password

    }

    return this.http.post(this.rootUrl + '/login', body);

  }

}

**Register.service.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class RegisterService {

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

  constructor(private http: HttpClient) { }

  insertUser(firstName: string, lastName: string,userName: string, password: string, dob:Date, phone: number, address: string, identityType: string,identity:string, email: string) {

    var body = {

      fname : firstName,

      lname : lastName,

      username: userName,

      password: password,

      dob:dob,

      phone : phone,

      address : address,

      identityType:identityType,

      identity:identity,

      email : email,

    }

    return this.http.post(this.rootUrl + '/register', body);

  }

}

**Request.service.ts**

import { Injectable } from '@angular/core';

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

import {ChequebookResponse} from './\_models/chequebookresponse'

@Injectable({

  providedIn: 'root'

})

export class RequestService {

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

  constructor(private http: HttpClient) { }

  insertRequest(accNo: number,pages:number=20) {

    var body = {

      account: accNo,

      no\_of\_pages: pages,

    }

    console.log(body);

    return this.http.post<ChequebookResponse>(this.rootUrl + '/cheque/request', body);

  }

}

**Transaction.service.ts**

import { Injectable } from '@angular/core';

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

import { Transaction } from './\_models/transaction'

import { SavingAccount } from './\_models/savingaccount'

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class TransactionService {

  private url:String;

  constructor(private http:HttpClient) {

    this.url="http://localhost:8080"

   }

   public getTransactions(accNo):Observable<Transaction[]>{

    return this.http.get<Transaction[]>(this.url+"/account/getHistory/"+accNo);

  }

  public getSavingAccount(username):Observable<SavingAccount>{

    return this.http.get<SavingAccount>(this.url+"/account/getsaving/"+username);

  }

}

**Transfer.service.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class TransferService {

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

    constructor(private http: HttpClient) { }

    insertEntry(username:string, saccount:string,ifscNo:string,raccount:string,amount:number) {

      var body = {

        username:username,

        saccount: saccount,

        ifsc: ifscNo,

        raccount:raccount,

        amount:amount

      }

      console.log(body);

      return this.http.post(this.rootUrl + '/account/transfer', body);

    }

  }

**transferhistory.service.ts**

import { Injectable } from '@angular/core';

import {TransferHistory} from './\_models/transferhistory';

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class TransferhistoryService {

  private url:String;

  constructor(private http:HttpClient) {

    this.url="http://localhost:8080"

   }

   public getTransferHistory(accNo):Observable<TransferHistory[]>{

    return this.http.get<TransferHistory[]>(this.url+"/account/getTransfers/"+accNo);

  }

  // public getSavingAccount(username):Observable<SavingAccount>{

  //   return this.http.get<SavingAccount>(this.url+"/account/getsaving/"+username);

  // }

}

**Update.service.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class UpdateService {

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

  constructor(private http: HttpClient) { }

  update(username:string,phone: number,email: string,address: string,prevpassword:string,newpassword:string) {

    var body = {

      username:username,

      phone : phone,

      email: email,

      address : address,

      password: prevpassword,

      newpassword:newpassword

    }

    console.log(body);

    return this.http.put(this.rootUrl + '/profile/update', body);

  }

}

**User.service.ts**

import { Injectable } from '@angular/core';

import { UserDisplay } from './\_models/userdisplay';

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

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class UserService {

  private url:string;

  constructor(private http:HttpClient) {

    this.url="http://localhost:8080"

   }

   public getUser(username):Observable<UserDisplay>{

    return this.http.get<UserDisplay>(this.url+"/home/"+username);

  }

}

**Frontend Admin Portal:**

**Autorhizationservice.ts**

import { Injectable } from '@angular/core';

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

import { Observable } from 'rxjs';

import { AuthorizeUser } from './model/authorizeUser';

@Injectable({

  providedIn: 'root'

})

export class AuthorizationService {

  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) { }

  getRequestData(){

    return this.http.get<AuthorizeUser[]>(this.rootUrl + '/unauthorized/all');

  }

  authorizeAccount(username){

    return this.http.get(this.rootUrl + username + '/authorize');

  }

  rejectRequest(username){

    return this.http.get(this.rootUrl + username + '/authorize/cancel' );

  }

}

**Checkbookservice.ts**

import { Injectable } from '@angular/core';

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

import { Observable } from 'rxjs';

import { CheckbookRequest } from './model/checkbookRequest';

@Injectable({

  providedIn: 'root'

})

export class CheckbookService {

  //readonly rootUrl = 'localhost:<port>/user/{username}/chequebook/request/confirm';

  readonly rootUrl = 'http://localhost:8084/user/';

  //readonly dataUrl= 'localhost:<port>/chequebook/request/all';

  readonly dataUrl= 'http://localhost:8084/chequebook/request/all';

  private data:any=[]

constructor(private http: HttpClient) { }

confirmCheckbookService(account){

  return this.http.get(this.rootUrl + account + '/chequebook/request/confirm');

}

getRequestsData():Observable<CheckbookRequest[]> {

  return this.http.get<CheckbookRequest[]>(this.dataUrl);

}

}

**Disableservice.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class DisableService {

  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) {

  }

  disableLoginService(username){

    return this.http.get(this.rootUrl + username + '/disable');

  }

  }

**Enableservice.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class EnableService {

  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) {

  }

  enableLoginService(username){

    return this.http.get(this.rootUrl + username + '/enable');

  }

}

**Feature.service.ts**

import { Injectable } from '@angular/core';

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

@Injectable({

  providedIn: 'root'

})

export class FeaturesService {

  id : number

  readonly rootUrl = 'http://localhost:8084/user/';

  constructor(private http: HttpClient) {

  }

  setFeatures(username,value){

    this.id=value

    console.log(this.id)

    return this.http.get(this.rootUrl + username + '/features/' + value);

  }

}

**User.service.ts**

import { Injectable } from '@angular/core';

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

import { UserData } from './model/userData';

import { Observable } from 'rxjs';

@Injectable({

  providedIn: 'root'

})

export class UsersService {

  readonly url = 'http://localhost:8084/';

  constructor(private http:HttpClient) {

   }

   public getAllUsers(){

    return this.http.get<any[]>(this.url+"user/all");

  }

}

**Backend Amin Portal:**

**Pom.xml**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.3.2.RELEASE</version>

        <relativePath/> <!-- lookup parent from repository -->

    </parent>

    <groupId>com.admin</groupId>

    <artifactId>admin\_service</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>admin\_service</name>

    <description>Admin service for icin bank</description>

    <properties>

        <java.version>1.8</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-jpa</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

            <exclusions>

                <exclusion>

                    <groupId>org.junit.vintage</groupId>

                    <artifactId>junit-vintage-engine</artifactId>

                </exclusion>

            </exclusions>

        </dependency>

        <dependency>

             <groupId>mysql</groupId>

             <artifactId>mysql-connector-java</artifactId>

        </dependency>

        <dependency>

          <groupId>org.apache.commons</groupId>

          <artifactId>commons-email</artifactId>

          <version>1.5</version>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>

**Admin.controller.java**

package com.admin.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.admin.model.ChequebookRequest;

import com.admin.model.User;

import com.admin.model.UserDisplay;

import com.admin.service.impl.AdminServiceImpl;

import com.admin.service.impl.MailServiceImpl;

@RestController

@CrossOrigin

public class AdminController {

    @Autowired

    private AdminServiceImpl service;

    @Autowired

    private MailServiceImpl 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("/user/transfers/{account}")

//  public List<Transfer> getTransactionList(@PathVariable("account") int account)

//  {

//      return service.getAllTransactions(account);

//  }

    @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);

    }

}

**Account repository.java**

package com.admin.dao;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.admin.model.Account;

@Repository

public interface AccountRepository extends CrudRepository<Account, Integer>{

    public Account findByUsername(String username);

    public Account findByAccno(long accno);

}

**Checkbookrequestrepository.java**

package com.admin.dao;

import java.util.List;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.transaction.annotation.Transactional;

import org.springframework.data.jpa.repository.Query;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.admin.model.ChequebookRequest;

@Repository

public interface ChequeBookRequestsRepository extends CrudRepository<ChequebookRequest, Integer>{

    @Modifying

    @Transactional

    @Query("update ChequebookRequest c set c.requestStatus=1 where c.account = ?1")

    void setChequebookInfoByAccount(long accNo);

    @Query("FROM ChequebookRequest c where c.requestStatus=FALSE")

    public List<ChequebookRequest> findAllChequebookRequests();

}

**Saccountrepository.java**

package com.admin.dao;

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

import org.springframework.stereotype.Repository;

import com.admin.model.Saccount;

@Repository

public interface SaccountRepository extends JpaRepository<Saccount, Integer>{

    public Saccount findByAccno(long accNo);

}

**Transferrepository.java**

package com.admin.dao;

import java.util.List;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import com.admin.model.Transfer;

@Repository

public interface TransferRepository extends CrudRepository<Transfer, Integer> {

    public List<Transfer> findBySaccount(long saccount);

    public List<Transfer> findByRaccount(long racoount);

}

**UserDisplayRepository.java**

package com.admin.dao;

import java.util.List;

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

import org.springframework.data.jpa.repository.Query;

import com.admin.model.User;

import com.admin.model.UserDisplay;

public interface UserDisplayRepository extends JpaRepository<User, Integer>{

    @Query("SELECT new com.admin.model.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.admin.model.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.admin.model.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);

}

**Userrepository.java**

package com.admin.dao;

import org.springframework.data.repository.CrudRepository;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.admin.model.User;

import java.util.List;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

@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=1 where u.username = ?1")

    void disableUser(String username);

    @Modifying

    @Transactional

    @Query("update User u set u.status=0 where u.username = ?1")

    void enableUser(String username);

    @Modifying

    @Transactional

    @Query("update User u set u.authorizationStatus=1 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);

}

**Account.java**

package com.admin.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.persistence.Table;

@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;

    }

}

**Checkbookrequest.java**

package com.admin.model;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.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 long getAccount() {

        return account;

    }

    public void setAccount(long account) {

        this.account = account;

    }

    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;

    }

    public String getAccType() {

        return accType;

    }

    public void setAccType(String accType) {

        this.accType = accType;

    }

    public boolean getRequestStatus() {

        return requestStatus;

    }

    public void setRequestStatus(boolean requestStatus) {

        this.requestStatus = requestStatus;

    }

    public int getId() {

        return id;

    }

    public void setId(int id) {

        this.id = id;

    }

}

**Saccount.java**

package com.admin.model;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.OneToOne;

import javax.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;

    }

}

**Transfer.java**

package com.admin.model;

import java.time.LocalDate;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.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;

    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;

    }

    @Override

    public int compareTo(Transfer o) {

        Integer i1=this.id;

        Integer i2=o.id;

        return i2.compareTo(i1);

    }

}

**User.java**

package com.admin.model;

import java.sql.Date;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import com.fasterxml.jackson.annotation.JsonFormat;

@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 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 int getFeatureStatus() {

        return featureStatus;

    }

    public void setFeatureStatus(int featureStatus) {

        this.featureStatus = featureStatus;

    }

    public boolean isStatus() {

        return status;

    }

    public boolean isAuthorizationStatus() {

        return authorizationStatus;

    }

    public void setAuthorizationStatus(boolean authorizationStatus) {

        this.authorizationStatus = authorizationStatus;

    }

    public void setId(int id) {

        this.id = id;

    }

    public void setStatus(boolean status) {

        this.status = status;

    }

    public Integer getId() {

        return id;

    }

    public void setId(Integer 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;

    }

}

**UserDisplay.java**

package com.admin.model;

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() {

    }

    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 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 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 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;

    }

    public int getFeatureStatus() {

        return featureStatus;

    }

    public void setFeatureStatus(int featureStatus) {

        this.featureStatus = featureStatus;

    }

}

**Adminserviceapplication.java**

package com.admin;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class AdminServiceApplication {

    public static void main(String[] args) {

        SpringApplication.run(AdminServiceApplication.class, args);

    }

}

**Selenium Testing:**

**Admin automation.class**

package firsttestngpackage;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.\*;

import org.openqa.selenium.By.ByClassName;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.Assert;

import org.testng.annotations.\*;

public class AdminAutomation {

  public String baseUrl = "http://localhost:4201";

    //String driverPath = "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe";

   String driverPath = "D:\\geckodriver\\geckodriver.exe";

    public WebDriver driver ;

  @BeforeTest

  public void launchBrowser() {

      System.out.println("Launching Firefox Browser");

      System.setProperty("webdriver.gecko.driver", driverPath);

      driver = new FirefoxDriver();

      driver.get(baseUrl);

      driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

  }

@Test(priority=0) public void login\_Pass() {

     driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

     driver.findElement(By.name("inputUserName")).sendKeys("madhuri");

      driver.findElement(By.name("password")).sendKeys("mad12345");

      //Login Button

      driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/button")).click();

      String actualUrl="http://localhost:4200/user-account";

      String expectedUrl= driver.getCurrentUrl();

      if(actualUrl.equalsIgnoreCase(expectedUrl)) {

      System.out.println("Login Successful"); }

      driver.manage().window().maximize();

}

@Test(priority=1) public void useraccount\_login\_enabling(){

  //User Account Hyperlink

  driver.findElement(By.xpath("/html/body/app-root/div/nav/div/ul/li[1]/a")).click();

  //Enable Button

  driver.findElement(By.xpath("/html/body/app-root/app-user-account/table/tbody/tr[1]/td[6]/button")).click();

  System.out.println("Enabled Login Feature");

  //Disable Button

  driver.findElement(By.xpath("/html/body/app-root/app-user-account/table/tbody/tr[2]/td[7]/button")).click();

  System.out.println("Disabled Login Feature");

  }

  @Test(priority=2) public void useraccount\_features(){

  //Click on the dropdown

  driver.findElement(By.xpath("/html/body/app-root/app-user-account/table/tbody/tr/td[9]/select")).click();

  //Select the first option

  driver.findElement(By.xpath("/html/body/app-root/app-user-account/table/tbody/tr/td[9]/select/option[1]")).click();

  //Set button

  driver.findElement(By.xpath("/html/body/app-root/app-user-account/table/tbody/tr/td[9]/button")).click();

  System.out.println("User Roles Changed");

  }

  @Test(priority=4) public void checkbookRequests() {

  //Checkbook Request Hyperlink

  driver.findElement(By.xpath("/html/body/app-root/div/nav/div/ul/li[2]/a")).click();

  //Confirm Request Button

  driver.findElement(By.xpath("/html/body/app-root/app-checkbook-requests/table/tbody/tr[1]/td[7]/button")).click();

  System.out.println("Request Confirmed");

  }

  @Test(priority=3) public void authorization() {

  //Authorization link

  driver.findElement(By.xpath("/html/body/app-root/div/nav/div/ul/li[3]/a")).click();

  //Create Account Button

  driver.findElement(By.xpath("/html/body/app-root/app-authorize-registration/table/tbody/tr/td[9]/button")).click();

  System.out.println("Authorized");

  //Cancel Button

  driver.findElement(By.xpath("/html/body/app-root/app-authorize-registration/table/tbody/tr[2]/td[10]/button")).click();

  System.out.println(" Not Authorized");

  }

  @Test(priority=5) public void logout() {

  //LogOut Button

  driver.findElement(By.xpath("/html/body/app-root/div/nav/div/ul/li[4]/a")).

  click(); System.out.println("Logged Out");

  }

  @Test(priority=6) public void login\_Fail() {

     driver.manage().timeouts().implicitlyWait(20, TimeUnit.SECONDS);

     driver.findElement(By.name("inputUserName")).sendKeys("madhuri");

      driver.findElement(By.name("password")).sendKeys("mad");

      //Login Button

      driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/button")).click();

      Alert alert = driver.switchTo().alert();

      alert.accept();

      String actualUrl="http://localhost:4200/user-account";

      String expectedUrl= driver.getCurrentUrl();

      if(!actualUrl.equalsIgnoreCase(expectedUrl)) {

      System.out.println("Login UnSuccessful"); }

      driver.manage().window().maximize();

}

}

**Login.java**

package Authentication;

import org.openqa.selenium.\*;

import org.openqa.selenium.chrome.\*;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import java.util.concurrent.TimeUnit;

import org.junit.Test;

public class Login {

  @Test

  public void LoginTest() throws InterruptedException {

    System.setProperty("webdriver.gecko.driver", "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe");

    WebDriver driver = new FirefoxDriver();

    driver.get("http://localhost:4200/login");

    Thread.sleep(5000);

    driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

    driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[1]/input")).sendKeys("Novina");

    driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[2]/input")).sendKeys("Novina123");

    driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[3]/button")).click();

    try

    {

        WebDriverWait wait=new WebDriverWait(driver, 14);

        wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("http://localhost:4200/home")));

        System.out.println("Login Successfull");

//      driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[5]/div/a[2]")).click();

//      wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

//      System.out.println("Signed Out");

    }

    catch(Exception e)

    {

        System.out.println("Error in browser!!\nPlease have a look");

    }

    Thread.sleep(5000);

    driver.quit();

  }

}

**Checkbookrequest.java**

package UserActions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class ChequeBookRequest {

    @Test

    public void chequeBookRequest() throws InterruptedException {

          System.setProperty("webdriver.gecko.driver", "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe");

            WebDriver driver = new FirefoxDriver();

            driver.get("http://localhost:4200/login");

            Thread.sleep(5000);

        driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

          driver.findElement(By.xpath(

          "/html/body/app-root/app-login/div/form/div[1]/input")).sendKeys("Novina");

          driver.findElement(By.xpath(

          "/html/body/app-root/app-login/div/form/div[2]/input")).sendKeys("Novina123")

          ;

          driver.findElement(By.xpath(

          "/html/body/app-root/app-login/div/form/div[3]/button")).click();

        try

        {

              WebDriverWait wait=new WebDriverWait(driver, 14);

              wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(

              "/html/body/app-root/app-home/div[1]/h2")));

              System.out.println("Login Successfull");

            driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[4]/a")).click();

            driver.findElement(By.xpath("/html/body/app-root/app-cheque-book-request/div/div[2]/select")).click();

            driver.findElement(By.xpath("/html/body/app-root/app-cheque-book-request/div/div[2]/select/option[1]")).click();

            driver.findElement(By.xpath("/html/body/app-root/app-cheque-book-request/div/form/button")).click();

            System.out.println("Cheque Book Issued Successfully!!");

//          driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[5]/div/a[2]")).click();

//          wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

//          System.out.println("Signed Out");

        }

        catch(Exception e)

        {

            System.out.println("Error in browser!!\nPlease have a look");

        }

        Thread.sleep(5000);

        driver.quit();

    }

}

**Editprofile.java**

package UserActions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class EditProfile {

    @Test

    public void EditPorfile() throws InterruptedException {

          System.setProperty("webdriver.gecko.driver",

          "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe"); WebDriver

          driver = new FirefoxDriver();

          driver.get("http://localhost:4200/home");

        /\*

         \* System.setProperty(

         \* "webdriver.chrome.driver","D:\\Novina\_BNP\\Simplilearn\_Projects\\Project 4\\Chrome Driver\\chromedriver.exe"

         \* ); WebDriver driver = new ChromeDriver();

         \*

         \* driver.get("http://localhost:4200/login");

         \*/

        Thread.sleep(5000);

        try

        {

            /\*

             \* WebDriverWait wait=new WebDriverWait(driver, 14);

             \* wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(

             \* "/html/body/app-root/app-home/div[1]/h2")));

             \* System.out.println("Login Successfull");

             \*/

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("//\*[@id=\"dropdown04\"]")).click();

            driver.findElement(By.xpath("/html/body/app-root/nav/div/ul/li[5]/div/a[1]")).click();

            driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[1]/input")).sendKeys("7666854389");

            driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[2]/input")).sendKeys("Mumbai");

            driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[2]/div[1]/input")).sendKeys("novinarayudu.97@gmail.com");

            driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[2]/div[2]/input")).sendKeys("123P1234");

            driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[2]/div[3]/input")).sendKeys("123P1234");

            //driver.findElement(By.xpath("/html/body/app-root/app-edit-profile/div[1]/form/div/div[3]/button")).click();

            System.out.println("Profile edited");

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("//\*[@id=\"dropdown04\"]")).click();

            driver.findElement(By.xpath("/html/body/app-root/nav/div/ul/li[5]/div/a[2]")).click();

            //wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

            System.out.println("Signed Out");

        }

        catch(Exception e)

        {

            System.out.println("Error in browser!!\nPlease have a look");

        }

        Thread.sleep(5000);

    }

}

**Register.java**

package UserActions;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class Register {

    @Test

    public void register() throws InterruptedException {

        System.setProperty("webdriver.chrome.driver", "D:\\Novina\_BNP\\Simplilearn\_Projects\\Project 4\\Chrome Driver\\chromedriver.exe");

        WebDriver driver = new ChromeDriver();

        driver.get("http://localhost:4200/login");

        Thread.sleep(5000);

        try {

            //WebDriverWait wait=new WebDriverWait(driver, 14);

            driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[3]/a")).click();

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[1]/input")).sendKeys("Novina");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[2]/input")).sendKeys("Rayudu");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[3]/input")).sendKeys("Novina");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[4]/input")).sendKeys("123456p");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[5]/input")).sendKeys("17/08/2020");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[6]/input")).sendKeys("7666854389");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[7]/input")).sendKeys("Mumbai");

            Select id=new Select(driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[8]/select")));

            id.selectByIndex(2);

            WebElement fileInput = driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[9]/input"));

            fileInput.sendKeys("D:\\dinos.png");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[10]/input")).sendKeys("abcde1234");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[11]/input")).sendKeys("novinarayudu.97@gmail,com");

            driver.findElement(By.xpath("/html/body/app-root/app-register/div/form/div/div[12]/button")).click();

            Thread.sleep(5000);

            //wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

            System.out.println("Registration Successfull");

        }

        catch(Exception e) {

            System.out.println("Erro in web browser\nPlease have a look");

        }

        Thread.sleep(5000);

        driver.quit();

    }

}

**Transactionhistory.java**

package UserActions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class TransactionHistory {

    @Test

    public void TransactionHistory() throws InterruptedException {

          System.setProperty("webdriver.gecko.driver", "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe");

            WebDriver driver = new FirefoxDriver();

            driver.get("http://localhost:4200/home");

        /\*

         \* System.setProperty("webdriver.chrome.driver",

         \* "D:\\Novina\_BNP\\Simplilearn\_Projects\\Project 4\\Chrome Driver\\chromedriver.exe"

         \* ); WebDriver driver = new ChromeDriver();

         \*

         \* driver.get("http://localhost:4200/login");

         \*/

        Thread.sleep(5000);

        /\*

         \* driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[1]/input")).sendKeys("Novina");

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[2]/input")).sendKeys("Novina123")

         \* ;

         \*

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[3]/button")).click();

         \*/

        try

        {

            /\*

             \* WebDriverWait wait=new WebDriverWait(driver, 14);

             \* wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(

             \* "/html/body/app-root/app-home/div[1]/h2")));

             \* System.out.println("Login Successfull");

             \*/

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[1]/a")).click();

            System.out.println("Transaction History Displayed");

            driver.findElement(By.xpath("//\*[@id=\"navbardrop\"]")).click();

            driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[5]/div/a[2]")).click();

           // wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

           System.out.println("Signed Out");

        }

        catch(Exception e)

        {

            System.out.println("Error in browser!!\nPlease have a look");

        }

        Thread.sleep(5000);

        driver.quit();

    }

}

**Transferhistory.java**

package UserActions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox.FirefoxDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class TransferHistory {

    @Test

    public void TransactionHistory() throws InterruptedException {

        /\*

         \* System.setProperty("webdriver.chrome.driver",

         \* "D:\\Novina\_BNP\\Simplilearn\_Projects\\Project 4\\Chrome Driver\\chromedriver.exe"

         \* ); WebDriver driver = new ChromeDriver();

         \*

         \* driver.get("http://localhost:4200/login");

         \*/

         System.setProperty("webdriver.gecko.driver", "C:\\Users\\Nagaraj\\Downloads\\firefoxDriver\\geckodriver.exe");

            WebDriver driver = new FirefoxDriver();

            driver.get("http://localhost:4200/home");

        Thread.sleep(5000);

        driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

        /\*

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[1]/input")).sendKeys("Novina");

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[2]/input")).sendKeys("Novina123")

         \* ;

         \*

         \* driver.findElement(By.xpath(

         \* "/html/body/app-root/app-login/div/form/div[3]/button")).click();

         \*/

        try

        {

            /\*

             \* WebDriverWait wait=new WebDriverWait(driver, 14);

             \* wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath(

             \* "/html/body/app-root/app-home/div[1]/h2")));

             \* System.out.println("Login Successfull");

             \*/

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[2]/a")).click();

            System.out.println("Transfer History Displayed");

//          driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[5]/div/a[2]")).click();

//          wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

//          System.out.println("Signed Out");

        }

        catch(Exception e)

        {

            System.out.println("Error in browser!!\nPlease have a look");

        }

        Thread.sleep(5000);

        driver.quit();

    }

}

**Transfermoney.java**

package UserActions;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.support.ui.ExpectedConditions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.testng.annotations.Test;

public class TransferMoney {

    @Test

    public void TransferMoney() throws InterruptedException{

        System.setProperty("webdriver.chrome.driver", "D:\\Novina\_BNP\\Simplilearn\_Projects\\Project 4\\Chrome Driver\\chromedriver.exe");

        WebDriver driver = new ChromeDriver();

        driver.get("http://localhost:4200/login");

        Thread.sleep(5000);

        driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

        driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[1]/input")).sendKeys("Novina");

        driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[2]/input")).sendKeys("Novina123");

        driver.findElement(By.xpath("/html/body/app-root/app-login/div/form/div[3]/button")).click();

        try

        {

            WebDriverWait wait=new WebDriverWait(driver, 14);

            wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-home/div[1]/h2")));

            System.out.println("Login Successfull");

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[3]/a")).click();

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("/html/body/app-root/app-transfer-between-accounts/div/form/div[3]/input")).sendKeys("123456789");

            driver.findElement(By.xpath("/html/body/app-root/app-transfer-between-accounts/div/form/div[4]/input")).sendKeys("1234");

            driver.findElement(By.xpath("/html/body/app-root/app-transfer-between-accounts/div/form/div[5]/input")).sendKeys("1000");

            Thread.sleep(5000);

            driver.manage().timeouts().implicitlyWait(14, TimeUnit.SECONDS);

            driver.findElement(By.xpath("/html/body/app-root/app-transfer-between-accounts/div/form/div[6]/button")).click();

            wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-home/div[1]/h2")));

            System.out.println("Transfer Money Successfull");

//          driver.findElement(By.xpath("/html/body/app-root/nav/ul/li[5]/div/a[2]")).click();

//          wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("/html/body/app-root/app-login/div/form/h3")));

//          System.out.println("Signed Out");

        }

        catch(Exception e)

        {

            System.out.println("Error in browser!!\nPlease have a look");

        }

        Thread.sleep(5000);

        driver.quit();

    }

}

**Pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>UserPortalTesting</groupId>

  <artifactId>UserPortalTesting</artifactId>

  <version>0.0.1-SNAPSHOT</version>

  <name>UserPortalAuthentication</name>

  <description>User portal testing</description>

  <dependencies>

    <dependency>

        <groupId>org.seleniumhq.selenium</groupId>

        <artifactId>selenium-java</artifactId>

        <version>3.141.59</version>

    </dependency>

    <dependency>

        <groupId>junit</groupId>

        <artifactId>junit</artifactId>

        <version>4.12</version>

        <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-api</artifactId>

        <version>1.7.5</version>

    </dependency>

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-log4j12</artifactId>

        <version>1.7.5</version>

    </dependency>

  </dependencies>

  </project>