**Controller**:

package com.icin.app.controller;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.http.ResponseEntity.BodyBuilder;

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

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

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

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

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

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

import com.icin.app.model.Admin;

import com.icin.app.model.Customer;

import com.icin.app.model.Security;

import com.icin.app.model.Transaction;

import com.icin.app.model.Transfer;

import com.icin.app.service.AdminDAO;

import com.icin.app.service.AdminRepository;

import com.icin.app.service.CustomerDAO;

import com.icin.app.service.CustomerRepository;

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

@RestController

public class AdminController {

@Autowired

CustomerDAO customerRepo;

@Autowired

AdminDAO adminRepo;

@Autowired

Security loginCheck;

@GetMapping("/")

public String mainPage() {

return "Hello";

}

@PostMapping("/Registration")

public String Registration(@RequestParam String Userid, @RequestParam String password ) {

System.out.println(Userid+"\n" +password);

//call update from here

return "Registration";

}

@GetMapping("/Admin/blockuser")

public String Block(@RequestParam int id) {

return "block";

}

@PostMapping("/login")

public ResponseEntity<Security> Login(@RequestBody Admin admin ) {

if(adminRepo.checkLogin(admin)) {

loginCheck.setLogin(true);

return ResponseEntity.status(HttpStatus.ACCEPTED).body(loginCheck);

}

loginCheck.setLogin(false);

return ResponseEntity.status(HttpStatus.OK).body(loginCheck);

}

@PostMapping("/AuthorizeCheque")

public ResponseEntity<Customer> AuthorizeCheque(@RequestBody Customer customer) {

return ResponseEntity.status(HttpStatus.ACCEPTED).body(customerRepo.authorizeChequeRequest(customer.getId()));

}

@PostMapping("/customerSignUp")

public Customer customerSignUp(@RequestBody Customer customer ) {

return customerRepo.save(customer);

}

@GetMapping("/findAllAdmins")

public List<Admin> findAllAdmins() {

return adminRepo.findAll();

}

@GetMapping("/AllCustomersWithNoAccess")

public List<Customer> findAllCustomersWithNoAccess(){

return customerRepo.findAllWithNoAccess();

}

@GetMapping("/AllCustomers")

public List<Customer> findAllCustomers(){

return customerRepo.findAll();

}

@PostMapping("/blockCustomer")

public Customer block(@RequestBody Customer customer) {

customer.setBlockStatus(true);

return customerRepo.update(customer);

}

@GetMapping("/getCustomerById")

public Customer getCustomerById(@RequestParam int id) {

if(customerRepo.findById(id).isPresent()) {

return customerRepo.findById(id).get();

}

return null;

}

@PostMapping("/customerLogin")

public Security customerLogin(@RequestBody Customer customer) {

return customerRepo.LoginCheck(customer);

}

@PostMapping("/requestChequeBook")

public Customer customerChequeRequest(@RequestParam int id) {

return customerRepo.chequeRequest(id);

}

@PostMapping("/Transfer")

public Transfer customerChequeRequest(@RequestBody Transfer transfer) {

return customerRepo.transferRequest(transfer) ;

}

@PostMapping("/transaction")

public Customer customerDeposit(@RequestBody Transaction transaction) {

return customerRepo.transactionService(transaction);

}

@PostMapping("/AuthorizeAccess")

public ResponseEntity<Customer> giveAccess(@RequestBody Customer customer){

return ResponseEntity.status(HttpStatus.ACCEPTED).body(customerRepo.giveAccess(customer.getId()));

}

@GetMapping("/testTransfer")

public Transfer test() {

Transfer transaction = new Transfer();

transaction.setAmount(2000);

transaction.setCustomerId(1);

transaction.setBeneficiaryId(1213141516);

return transaction;

}

}

**Application**:

package com.icin.app;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class IcinBankApplication {

public static void main(String[] args) {

SpringApplication.run(IcinBankApplication.class, args);

}

}

**Admin Model:**

package com.icin.app.model;

import javax.persistence.Entity;

import javax.persistence.Table;

import javax.persistence.Id;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Table(name="admindetails")

public class Admin {

private String employeename;

private String userName;

@Id

private int adminid;

private String password;

public String getEmployeename() {

return employeename;

}

public void setEmployeename(String employeename) {

this.employeename = employeename;

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public int getAdminid() {

return adminid;

}

public void setAdminid(int adminid) {

this.adminid = adminid;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

public Admin() {

super();

}

public Admin(String employeename, String userName, String password) {

this.employeename= employeename;

this.userName = userName;

this.password = password;

}

public Admin(String userName, String password) {

this.userName=userName;

this.password=password;

}

}

**Customer Model:**

package com.icin.app.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import lombok.Data;

import lombok.NoArgsConstructor;

@Entity

@Table(name="customerdetails")

public class Customer {

@Id

private int id;

private String userName;

private String firstName;

private String lastName;

private String pasword;

private String email;

private String phoneNumber;

private boolean access;

private boolean blockStatus;

private boolean chequeBookRequest;

private double savingsAmount;

private double primaryAmount;

private double accountNumber;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public Customer() {

super();

}

@Override

public String toString() {

return "Customer [id=" + id + ", userName=" + userName + ", firstName=" + firstName + ", lastName=" + lastName

+ ", pasword=" + pasword + ", email=" + email + ", phoneNumber=" + phoneNumber + ", access=" + access

+ ", blockStatus=" + blockStatus + ", chequeBookRequest=" + chequeBookRequest + ", savingsAmount="

+ savingsAmount + ", primaryAmount=" + primaryAmount + ", accountNumber=" + accountNumber + "]";

}

public String getUserName() {

return userName;

}

public void setUserName(String userName) {

this.userName = userName;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

public String getpasword() {

return pasword;

}

public void setpasword(String pasword) {

this.pasword = pasword;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public String getPhoneNumber() {

return phoneNumber;

}

public void setPhoneNumber(String phoneNumber) {

this.phoneNumber = phoneNumber;

}

public boolean isAccess() {

return access;

}

public void setAccess(boolean access) {

this.access = access;

}

public boolean isBlockStatus() {

return blockStatus;

}

public void setBlockStatus(boolean blockStatus) {

this.blockStatus = blockStatus;

}

public boolean isChequeBookRequest() {

return chequeBookRequest;

}

public void setChequeBookRequest(boolean chequeBookRequest) {

this.chequeBookRequest = chequeBookRequest;

}

public double getSavingsAmount() {

return savingsAmount;

}

public void setSavingsAmount(double savingsAmount) {

this.savingsAmount = savingsAmount;

}

public double getPrimaryAmount() {

return primaryAmount;

}

public void setPrimaryAmount(double primaryAmount) {

this.primaryAmount = primaryAmount;

}

public double getAccountNumber() {

return accountNumber;

}

public void setAccountNumber(double accountNumber) {

this.accountNumber = accountNumber;

}

public Customer(String userName, String pasword) {

super();

this.userName = userName;

this.pasword = pasword;

}

public Customer(String firstName, String lastName, String phonenumber, String email,String userName, String pasword, double savingsAmount, double primaryAmount, double accountNumber) {

this.firstName = firstName;

this.email = email;

this.lastName = lastName;

this.userName = userName;

this.pasword = pasword;

this.phoneNumber =phonenumber;

this.savingsAmount = savingsAmount;

this.primaryAmount = primaryAmount;

this.access =false;

this.blockStatus = false;

this.chequeBookRequest=false;

this.accountNumber= accountNumber;

}

}

**Security Model**

**package** com.icin.app.model;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Security {

**boolean** login;

**protected** **int** id;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** **boolean** isLogin() {

**return** login;

}

**public** **void** setLogin(**boolean** login) {

**this**.login = login;

}

**public** Security() {

**super**();

}

**public** Security(**boolean** login) {

**super**();

**this**.login = login;

}

**public** Security(**boolean** login, **int** id) {

**this**.login= login;

**this**.id = id;

}

}

**Transaction Model**

**package** com.icin.app.model;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Transaction {

**private** **int** customerId;

**private** **double** amount;

**private** Boolean isDeposit;

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** Boolean getIsDeposit() {

**return** isDeposit;

}

**public** **void** setIsDeposit(Boolean isDeposit) {

**this**.isDeposit = isDeposit;

}

**public** Transaction(**int** customerId, **double** amount, Boolean isDeposit) {

**super**();

**this**.customerId = customerId;

**this**.amount = amount;

**this**.isDeposit = isDeposit;

}

**public** Transaction() {

**super**();

}

}

**Transfer Model**

**package** com.icin.app.model;

**import** org.springframework.stereotype.Component;

@Component

**public** **class** Transfer{

**private** **int** CustomerId;

**public** **int** getCustomerId() {

**return** CustomerId;

}

**public** **void** setCustomerId(**int** customerId) {

CustomerId = customerId;

}

**private** **double** beneficiaryId;

**private** String beneficiaryName;

**private** **double** amount;

**public** **double** getAmount() {

**return** amount;

}

**public** **void** setAmount(**double** amount) {

**this**.amount = amount;

}

**public** Transfer(**int** customerId, **double** beneficiaryId, String beneficiaryName, Boolean transactionStatus, **double** amount) {

**super**();

CustomerId = customerId;

**this**.beneficiaryId = beneficiaryId;

**this**.beneficiaryName = beneficiaryName;

**this**.amount = amount;

}

**public** **double** getBeneficiaryId() {

**return** beneficiaryId;

}

**public** **void** setBeneficiaryId(**double** beneficiaryId) {

**this**.beneficiaryId = beneficiaryId;

}

**public** String getBeneficiaryName() {

**return** beneficiaryName;

}

**public** **void** setBeneficiaryName(String beneficiaryName) {

**this**.beneficiaryName = beneficiaryName;

}

**public** Transfer(**double** beneficiaryId, String beneficiaryName ) {

**super**();

**this**.beneficiaryId = beneficiaryId;

**this**.beneficiaryName = beneficiaryName;

}

**public** Transfer() {

**super**();

}

**public** Transfer(**int** customerId, **double** beneficiaryId, String beneficiaryName,

**double** amount) {

**super**();

**this**.CustomerId = customerId;

**this**.beneficiaryId = beneficiaryId;

**this**.beneficiaryName = beneficiaryName;

**this**.amount = amount;

}

}

**ADMIN DAO**

package com.icin.app.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import com.icin.app.model.Admin;

import com.icin.app.model.Customer;

@Service

public class AdminDAO{

@Autowired

AdminRepository adminRepository;

public List<Admin> findByUserName(String name) {

return adminRepository.findAdminByUsername(name) ;

}

public Admin findByemployeeName() {

return null;

}

public List<Admin> findAll(){

return adminRepository.findAll();

}

public boolean checkLogin(Admin admin) {

List<Admin> l =findByUserName(admin.getUserName());

for(Admin a : l) {

if(a.getPassword().equals(admin.getPassword())) {

return true;

}

}

return false;

}

}

**Admin Repository:**

package com.icin.app.service;

import java.util.List;

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

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

import org.springframework.stereotype.Repository;

import com.icin.app.model.Admin;

public interface AdminRepository extends JpaRepository<Admin, Integer> {

public List<Admin> findAdminByEmployeename(String employeename);

@Query("select admin from Admin admin where admin.adminid=?1")

public Admin findAdminById(int id);

@Query(value="select \* from admindetails a where a.user\_name = ?1", nativeQuery=true)

public List<Admin> findAdminByUsername(String userName);

}

**Customer DAO**

package com.icin.app.service;

import java.util.List;

import java.util.Optional;

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

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

import org.springframework.stereotype.Service;

import com.icin.app.model.Customer;

import com.icin.app.model.Security;

import com.icin.app.model.Transaction;

import com.icin.app.model.Transfer;

@Service

public class CustomerDAO {

@Autowired

CustomerRepository customerRepo;

@Autowired

Security security;

public Customer save(Customer customer) {

return customerRepo.save(customer);

}

public List<Customer> findAll(){

return customerRepo.findAll();

}

public List<Customer> findAllWithNoAccess(){

return customerRepo.findAllWithNoAccess();

}

public Customer update(Customer customer) {

return customerRepo.save(customer);

}

public Optional<Customer> findById(int id) {

return customerRepo.findById(id);

}

public Security LoginCheck(Customer customer) {

Customer customerFullObj = customerRepo.findByUserName(customer.getUserName());

System.out.println(customerFullObj.toString());

if(customerFullObj.isAccess()&& !customerFullObj.isBlockStatus() && customerFullObj.getpasword().equals(customer.getpasword())) {

security.setLogin(true);

security.setId(customerFullObj.getId());

return security;

}

security.setLogin(false);

return security;

}

public Customer chequeRequest(int id) {

Optional<Customer> customerFullObj = customerRepo.findById(id);

if(customerFullObj.isPresent()) {

customerFullObj.get().setChequeBookRequest(true);

return customerRepo.save(customerFullObj.get());

}

return null;

}

public Customer authorizeChequeRequest(int id) {

Optional<Customer> customerFullObj = customerRepo.findById(id);

if(customerFullObj.isPresent()) {

customerFullObj.get().setChequeBookRequest(false);

return customerRepo.save(customerFullObj.get());

}

return null;

}

public Transfer transferRequest(Transfer transfer){

Optional<Customer> customerObj = customerRepo.findById(transfer.getCustomerId());

Customer benificiary = customerRepo.findByAccountNumber(transfer.getBeneficiaryId());

if(customerObj.isPresent()) {

customerObj.get().setPrimaryAmount(customerObj.get().getPrimaryAmount()- transfer.getAmount());

customerRepo.save(customerObj.get());

}

benificiary.setPrimaryAmount(benificiary.getPrimaryAmount()+transfer.getAmount());

customerRepo.save(benificiary);

return transfer;

}

public Customer transactionService(Transaction transaction) {

Optional<Customer> customerObj = customerRepo.findById(transaction.getCustomerId());

if(customerObj.isPresent()) {

if(transaction.getIsDeposit()) {

customerObj.get().setPrimaryAmount(customerObj.get().getPrimaryAmount()+transaction.getAmount());

}

else {

customerObj.get().setPrimaryAmount(customerObj.get().getPrimaryAmount()-transaction.getAmount());

}

}

return customerRepo.save(customerObj.get());

}

public Customer giveAccess(int id) {

Optional<Customer> customerObj = customerRepo.findById(id);

if(customerObj.isPresent()) {

customerObj.get().setAccess(true);

}

return customerRepo.save(customerObj.get());

}

}

**Customer Repository**

package com.icin.app.service;

import java.util.List;

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

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

import com.icin.app.model.Customer;

public interface CustomerRepository extends JpaRepository<Customer, Integer> {

@Query(value="select \* from customerdetails c where c.\"firstName\"=(?1)", nativeQuery=true)

public Customer findByName(String name);

// @Query(value="INSERT into ")

// public Customer save(Customer customer);

@Query(value="select \* from customerdetails", nativeQuery=true)

public List<Customer> findAll();

@Query(value="select \* from customerdetails c where c.access=false", nativeQuery=true)

public List<Customer> findAllWithNoAccess();

@Query(value="select \* from customerdetails c where c.user\_name=(?1)", nativeQuery=true)

public Customer findByUserName(String userName);

@Query(value="select \* from customerdetails c where c.account\_number= (?1)", nativeQuery= true)

public Customer findByAccountNumber(double accountNumber);}

**Application Properties:**

server.port = 8085

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/icinbank

spring.datasource.username= root

spring.datasource.password= 98765

**Application-test Properties:**

spring.datasource.url= jdbc:h2:mem:icinBank;

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username= sri

spring.datasource.password= root

spring.jpa.database.platform = org.hibernate.dialect.H2Dialect

spring.sql.init.data-locations= classpath:data-h2.sql

spring.cloud.discovery.enabled = false

spring.cloud.config.discovery.enabled= false

**Application Test**

package com.icin.app;

import static org.assertj.core.api.Assertions.assertThat;

import java.util.Optional;

import org.junit.jupiter.api.Test;

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

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.test.context.TestPropertySource;

import com.icin.app.model.Customer;

import com.icin.app.service.AdminDAO;

import com.icin.app.service.CustomerDAO;

@SpringBootTest

@TestPropertySource(locations= "classpath:application-test.properties")

class IcinBankApplicationTests {

@Autowired

AdminDAO adminRepo;

@Autowired

CustomerDAO customerRepo;

@Test

void contextLoads() {

}

@Test

void SaveToCustomerDb() {

Customer actual = new Customer("firstName","lastName","1214151618","email@email","test","test",1500,1200,1122334455);

customerRepo.save(actual);

Optional<Customer> expected = customerRepo.findById(1);

assertThat(expected.get().equals(actual));

}

@Test

void giveAccess() {

customerRepo.giveAccess(1);

Optional<Customer> expected = customerRepo.findById(1);

assertThat(expected.get().isAccess()==true);

}

@Test

void RequestCheque() {

customerRepo.chequeRequest(1);

Optional<Customer> expected = customerRepo.findById(1);

assertThat(expected.get().isChequeBookRequest()==true);

}

@Test

void IssueChequeReuest() {

customerRepo.authorizeChequeRequest(1);

Optional<Customer> expected = customerRepo.findById(1);

assertThat(expected.get().isChequeBookRequest()==false);;

}

}

**Admin Login.css**

/\* Bordered form \*/

form {

  border: 3px solid #f1f1f1;

}

/\* Full-width inputs \*/

input[type="text"],

input[type="password"] {

  width: 25%;

  padding: 12px 20px;

  margin-left: 15%;

  margin-right: 60%;

  margin-bottom: 8px;

  display: inline-block;

  border: 1px solid #ccc;

  box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

  background-color: #000503;

  color: white;

  padding: 14px 20px;

  margin: 8px 0;

  border: none;

  cursor: pointer;

  width: 20%;

}

/\* Add a hover effect for buttons \*/

button:hover {

  opacity: 0.8;

}

/\* Extra style for the cancel button (red) \*/

.cancelbtn {

  width: auto;

  padding: 10px 18px;

  background-color: #f44336;

}

/\* Center the avatar image inside this container \*/

.imgcontainer {

  text-align: center;

  margin: 24px 0 12px 0;

}

/\* Avatar image \*/

img.avatar {

  width: 40%;

  border-radius: 50%;

}

/\* Add padding to containers \*/

.container {

  padding: 16px;

}

/\* The "Forgot password" text \*/

span.psw {

  float: right;

  padding-top: 16px;

}

/\* labels of text fields \*/

.container label {

  margin-left: 12.5%;

}

/\* login button \*/

.loginbtn {

  margin-left: 17%;

}

h2 {

  color: grey;

}

/\* Change styles for span and cancel button on extra small screens \*/

@media screen and (max-width: 300px) {

  span.psw {

    display: block;

    float: none;

  }

  .cancelbtn {

    width: 100%;

  }

}

Admin-login.html

<form #adminLogin="ngForm" (ngSubmit)="onLogin(adminLogin.value)">

    <!-- <div class="imgcontainer">

      <img src="img\_avatar2.png" alt="Avatar" class="avatar">

    </div> -->

    <div class="container" >

        <h2>Admin Portal</h2>

      <label for="uname"><b>Username</b></label>

      <input type="text" placeholder="Enter Username" name="uname" required ngModel>

      <label for="psw"><b>Password</b></label>

      <input type="password" placeholder="Enter Password" name="psw" required ngModel>

      <button class="loginbtn" type="submit">Login</button>

      <label>

        <input type="checkbox" checked="checked" name="remember"> Remember me

      </label>

    </div>

    <div class="container" style="background-color:#f1f1f1">

      <button type="button" class="cancelbtn">Cancel</button>

      <span class="psw">Forgot <a href="#">password?</a></span>

    </div>

  </form>

Admin-loginSpec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdminLoginComponent } from './admin-login.component';

describe('AdminLoginComponent', () => {

  let component: AdminLoginComponent;

  let fixture: ComponentFixture<AdminLoginComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AdminLoginComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(AdminLoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

Admin Login.ts

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

import { ActivatedRoute, Router } from '@angular/router';

const loginObj = {

  userName: '',

  password: '',

};

@Component({

  selector: 'app-admin-login',

  templateUrl: './admin-login.component.html',

  styleUrls: ['./admin-login.component.css'],

})

export class AdminLoginComponent implements OnInit {

  constructor(

    private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router

  ) {}

  ngOnInit(): void {

    console.log('adminLogin component');

  }

  onLogin(result: { uname: string; psw: string }): void {

    console.log(result.uname);

    console.log(result.psw);

    loginObj.userName = result.uname;

    loginObj.password = result.psw;

    this.service.getLoginStatus(loginObj).subscribe((res) => {

      if (res.login) {

        localStorage.setItem("admin",loginObj.userName)

        this.router.navigate(['/Admin']);

      }

      else{

        alert("please check your credentials");

      }

    });

  }

}

Admin-Screen.css

#customers {

    font-family: Arial, Helvetica, sans-serif;

    border-collapse: collapse;

    width: 100%;

  }

  #customers td, #customers th {

    border: 1px solid #ddd;

    padding: 8px;

  }

  #customers tr:nth-child(even){background-color: #f2f2f2;}

  #customers tr:hover {background-color: #ddd;}

  #customers th {

    padding-top: 12px;

    padding-bottom: 12px;

    text-align: left;

    background-color: #0d0e0d;

    color: whitesmoke;

  }

  button{

    color: whitesmoke;

    background-color: rgb(44, 44, 44) ;

    border-radius: 2%;

  }

  button:hover{

    color:white;

    background-color: black;

    box-shadow: inset;

  }

  .righFloat{

    position: absolute;

    top:7%;

    right: 3%;

    border-radius: 5%;

    padding: 15px 32px;

    text-align: center;

    text-decoration: none;

    font-size: medium;

  }

  button:disabled,button[disabled]{

    cursor: not-allowed;

  }

Admin-Screen.html

<h2> Welcome Administrator</h2>

<button class="righFloat" (click)="logOut()"> Logout</button>

<table id="customers">

  <tr>

    <th>account number</th>

    <th>Block Status</th>

    <th>Block</th>

    <th>Access</th>

    <th>Provide Access</th>

    <th>Cheque Book Requested</th>

    <th>Issue ChequeBook</th>

    <tr \*ngFor="let customer of data;">

        <td>{{customer.accountNumber}}</td>

        <td>{{customer.blockStatus}}</td>

        <td><button [disabled]="customer.blockStatus===true" (click)="postBlock(customer)">Block</button></td>

        <td>{{customer.access}}</td>

        <td><button [disabled]="customer.access===true"(click)="provideAccess(customer)">Provide Access</button></td>

        <td>{{customer.chequeBookRequest}}</td>

        <td><button [disabled]="!customer.chequeBookRequest" (click)="provideChequeBook(customer)">Issue Cheque Book</button></td>

    </tr>

</table>

AdminScreenSpec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdminScreenComponent } from './admin-screen.component';

describe('AdminScreenComponent', () => {

  let component: AdminScreenComponent;

  let fixture: ComponentFixture<AdminScreenComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ AdminScreenComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(AdminScreenComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

AdminScreen.ts

import { getLocaleDateFormat } from '@angular/common';

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-admin-screen',

  templateUrl: './admin-screen.component.html',

  styleUrls: ['./admin-screen.component.css']

})

export class AdminScreenComponent implements OnInit {

  public data: any=[];

  constructor(private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router) { }

  ngOnInit(): void {

    if(localStorage.getItem("admin")){

      alert("Hello, you are logged in ");

      this.getData()

    }

    else{

      alert("Access forbidden you're being redirected to login page");

      this.router.navigate(["/Adminlogin"])

    }

  }

  getData(){

      this.service.getAllUsers().subscribe((res)=>{

        this.data = res;

        console.log(this.data);

      })

  }

  reloadPage(){

    window.location.reload();

  }

  postBlock(customer:any){

    this.service.postBlock(customer).subscribe((res)=>{

      console.log(res);

    })

    // console.log(customer);

    // this.reloadPage();

  }

  provideAccess(customer:any){

    this.service.postProvideAccess(customer).subscribe((res)=>{

      console.log(res);

    })

  }

  logOut(){

    localStorage.removeItem("admin");

    this.router.navigate(["/Adminlogin"])

  }

  provideChequeBook(customer:any){

    console.log(customer);

    this.service.postAuthorizeCheque(customer).subscribe((res)=>{

      if(res){

        alert("cheque book issued");

      }

    })

  }

}

Customer-Login.css

/\* Bordered form \*/

form {

  border: 3px solid #f1f1f1;

}

/\* Full-width inputs \*/

input[type="text"],

input[type="password"] {

  width: 25%;

  padding: 12px 20px;

  margin-top: 10px;

  margin-left: 15%;

  margin-right: 60%;

  margin-bottom: 8px;

  display: inline-block;

  border: 1px solid #ccc;

  box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

  background-color: #000503;

  color: white;

  padding: 14px 20px;

  margin: 8px 0;

  border: none;

  cursor: pointer;

  width: 20%;

}

/\* Add a hover effect for buttons \*/

button:hover {

  opacity: 0.8;

}

/\* Extra style for the cancel button (red) \*/

.cancelbtn {

  width: auto;

  padding: 10px 18px;

  background-color: #f44336;

}

/\* Center the avatar image inside this container \*/

.imgcontainer {

  text-align: center;

  margin: 24px 0 12px 0;

}

/\* Avatar image \*/

img.avatar {

  width: 40%;

  border-radius: 50%;

}

/\* Add padding to containers \*/

.container {

  padding: 16px;

}

/\* The "Forgot password" text \*/

span.psw {

  float: right;

  padding-top: 16px;

}

/\* labels of text fields \*/

.container label {

  margin-left: 12.5%;

}

/\* login button \*/

.loginbtn {

  margin-left: 17%;

}

/\* Change styles for span and cancel button on extra small screens \*/

@media screen and (max-width: 300px) {

  span.psw {

    display: block;

    float: none;

  }

  .cancelbtn {

    width: 100%;

  }

}

Customer-Login.html

<form #customerLogin="ngForm" (ngSubmit)="onLogin(customerLogin.value)">

    <div class="container" >

      <label for="cname"><b>Username</b></label>

      <input type="text" placeholder="Enter Username" name="cname" required ngModel>

      <label for="psw"><b>Password</b></label>

      <input type="password" placeholder="Enter Password" name="psw" required ngModel>

      <button class="loginbtn" type="submit">Login</button>

      <label>

        <input type="checkbox" checked="checked" name="remember"> Remember me

      </label>

    </div>

    <div class="container" style="background-color:#f1f1f1">

      <button type="button" class="cancelbtn">Cancel</button>

      <span class="psw">Forgot <a href="#">password?</a></span>

    </div>

  </form>

CustomerLoginSpect.Ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CustomerLoginComponent } from './customer-login.component';

describe('CustomerLoginComponent', () => {

  let component: CustomerLoginComponent;

  let fixture: ComponentFixture<CustomerLoginComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CustomerLoginComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(CustomerLoginComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

Customer-Login.Ts

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute, Router } from '@angular/router';

import { MatCardModule } from '@angular/material/card';

import { ServiceService } from '../service.service';

const customer={

  userName:"",

  pasword:""

}

@Component({

  selector: 'app-customer-login',

  templateUrl: './customer-login.component.html',

  styleUrls: ['./customer-login.component.css']

})

export class CustomerLoginComponent implements OnInit {

  constructor(

    private route: ActivatedRoute,

    private router: Router,

    private service: ServiceService

  ) { }

  ngOnInit(): void {

  }

  onLogin(result: { cname: any; psw: any; }){

    customer.userName = result.cname;

    customer.pasword =  result.psw;

    console.log(customer);

    this.loginCheck(customer);

  }

  loginCheck(obj: any):void{

    this.service.postCustomerLogin(obj).subscribe((res)=>{

      console.log(res);

      if(res.login){

        localStorage.setItem("CustomerId", res.id);

        this.router.navigate(['/icinBank']);

      }

      else{

        alert("Check your credentials or Wait for admin to provide you access")

      }

    })

  }

}

Customer-Main.css

.card {

  /\* Add shadows to create the "card" effect \*/

  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

  margin-top:5%;

  margin-right: 60%;

  margin-bottom: 2%;

  transition: 0.3s;

}

/\* On mouse-over, add a deeper shadow \*/

.card:hover {

  box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

}

/\* Add some padding inside the card container \*/

.container {

  padding: 16px 32px;

}

h4 {

  color: rgb(65, 65, 65);

}

.flex-container {

  display: flex;

  justify-content: space-between;

}

.float-card {

  position: absolute;

  top: 18%;

  right: 5%;

  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

  transition: 0.3s;

}

/\* .float-card:hover {

  box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

} \*/

.button {

  background-color: rgb(83, 83, 83);

  border: none;

  color: white;

  padding: 15px 32px;

  text-align: center;

  text-decoration: none;

  display: inline-block;

  font-size: 16px;

}

.button:hover {

  background-color: black;

}

.float-card2 {

  position: absolute;

  top: 18%;

  right: 25%;

  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

  transition: 0.3s;

}

.float-button{

  position: absolute;

  top: 5%;

  right: 5%;

}

Customer-Main.html

<div class="card">

    <div class="container">

        <div class="flex-container">

            <h4><b>Primary Balance: </b></h4>

            <h2><b>₹ {{data.primaryAmount}}</b></h2>

        </div>

    </div>

  </div>

<div class="card">

    <div class="container">

        <div class="flex-container">

            <h4><b>Savings Balance: </b></h4>

            <h2><b>₹ {{data.savingsAmount}}</b></h2>

        </div>

    </div>

</div>

<div class="card">

    <div class="container">

        <div class="flex-container">

            <button (click)="requestChequeBook()" class="button">Request Cheque Book</button>

            <button (click)="routeToTransferMoney()" class="button" >Transfer Money</button>

        </div>

    </div>

</div>

<div class="float-card">

    <div class="container">

        <div class="flex-container">

            <button (click)="routeToWithDraw()" class="button">Withdraw</button>

        </div>

    </div>

</div>

<div class="float-card2">

    <div class="container">

        <div class="flex-container">

            <button (click)="routeToDeposit()" class="button">Deposit</button>

        </div>

    </div>

</div>

<div class="float-button">

    <div class="container">

        <div class="flex-container">

            <button (click)="logoutHandler()" class="button">Log out</button>

        </div>

    </div>

</div>

Customer-Main-Spec.Ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CustomerMainScreenComponent } from './customer-main-screen.component';

describe('CustomerMainScreenComponent', () => {

  let component: CustomerMainScreenComponent;

  let fixture: ComponentFixture<CustomerMainScreenComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CustomerMainScreenComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(CustomerMainScreenComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

Customer-Main.Ts

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-customer-main-screen',

  templateUrl: './customer-main-screen.component.html',

  styleUrls: ['./customer-main-screen.component.css']

})

export class CustomerMainScreenComponent implements OnInit {

  public data: any=[];

  constructor(

    private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router

  ) { }

  ngOnInit(): void {

    if(localStorage.getItem("CustomerId")){

      this.service.getCustomerDataById(parseInt(localStorage.getItem("CustomerId")!)).subscribe(res =>{

        this.data= res

      })

    }

    else{

      alert("Access forbidden you're being redirected to login page");

      this.router.navigate(["/login"])

    }

  }

  logoutHandler():void{

    localStorage.removeItem("CustomerId");

    this.router.navigate(["/login"]);

  }

  requestChequeBook():void{

    this.service.postRequestChequeBook(parseInt(localStorage.getItem("CustomerId")!)).subscribe(res=>{

      if(res){

        alert("ChequeBook Requested Successfully")

      }

      else{

        alert("there was an error requesting ChequeBook, please try after some time")

      }

    });

  }

  routeToDeposit():void{

    this.router.navigate(["icinBank/Deposit"]);

  }

  routeToWithDraw():void{

    this.router.navigate(["icinBank/Withdraw"])

  }

  routeToTransferMoney():void{

    this.router.navigate(["icinBank/transfermoney"])

  }

}

Customer-SignUp.css

form {

  border: 3px solid #f1f1f1;

}

input[type="text"],

input[type="password"],

input[type="email"],

input[type="tel"] {

  width: 25%;

  padding: 12px 20px;

  margin-left: 15%;

  margin-right: 60%;

  margin-bottom: 8px;

  display: inline-block;

  border: 1px solid #ccc;

  box-sizing: border-box;

}

button {

  background-color: #000503;

  color: white;

  padding: 14px 20px;

  margin: 8px 0;

  border: none;

  cursor: pointer;

  width: 20%;

}

button:hover {

  opacity: 0.8;

}

.cancelbtn {

  width: auto;

  padding: 10px 18px;

  background-color: #f44336;

}

.imgcontainer {

  text-align: center;

  margin: 24px 0 12px 0;

}

img.avatar {

  width: 40%;

  border-radius: 50%;

}

.container {

  padding: 16px;

}

span.psw {

  float: right;

  padding-top: 16px;

}

.container label {

  margin-left: 12.5%;

}

/\* login button \*/

.loginbtn {

  margin-left: 17%;

}

@media screen and (max-width: 300px) {

  span.psw {

    display: block;

    float: none;

  }

  .cancelbtn {

    width: 100%;

  }

}

Customer-Signup.html

<form #customerSignUp="ngForm" (ngSubmit)="signUp(customerSignUp.value)" >

    <div class="container" >

      <label for="fname"><b>First Name</b></label>

      <input type="text" placeholder="Enter First Name" name="fname" required ngModel>

      <label for="lname"><b>Last Name</b></label>

      <input type="text" placeholder="Enter Last Name" name="lname" required ngModel>

      <label for="cname"><b>Username</b></label>

      <input type="text" placeholder="Enter Username" name="cname" required ngModel>

      <label for="mailid"><b>E-Mail</b></label>

      <input type="email" placeholder="Enter E-Mail Id" name="mailid" required ngModel>

      <label for="phoneno"><b>Phone no</b></label>

      <input type="tel" placeholder="Enter Phone no:" name="phoneno" required ngModel>

      <label for="psw"><b>Password</b></label>

      <input type="password" placeholder="Enter Password" name="psw" required ngModel>

      <label for="psw2"><b>Re-enter Password</b></label>

      <input type="password" placeholder="Re-Enter Password" name="psw2" required ngModel>

      <label for="accountNumber"><b>Bank Account Number</b></label>

      <input type="password" placeholder="Enter Bank Account Number" name="accountNumber" required ngModel>

      <label for="accountNumber2"><b>Bank Account Number</b></label>

      <input type="text" placeholder="Re-Enter Bank Account Number" name="accountNumber2" required ngModel>

      <button class="loginbtn" type="submit">Sign up</button>

    </div>

  </form>

Customer-SignUp-specs.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { CustomerSignUpComponent } from './customer-sign-up.component';

describe('CustomerSignUpComponent', () => {

  let component: CustomerSignUpComponent;

  let fixture: ComponentFixture<CustomerSignUpComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ CustomerSignUpComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(CustomerSignUpComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

Customer-Signup.ts

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

const signupObject = {

  firstName: '',

  lastName: '',

  phoneNumber: '',

  email: '',

  userName: '',

  pasword: '',

  savingsAmount: 1000,

  primaryAmount: 2000,

  accountNumber: 0,

};

@Component({

  selector: 'app-customer-sign-up',

  templateUrl: './customer-sign-up.component.html',

  styleUrls: ['./customer-sign-up.component.css'],

})

export class CustomerSignUpComponent implements OnInit {

  constructor(private service: ServiceService) {}

  ngOnInit(): void {}

  signUp(result: {

    fname: any;

    lname: any;

    cname: any;

    mailid: any;

    phoneno: any;

    psw: any;

    psw2: any;

    accountNumber: any;

    accountNumber2: any;

  }) {

    signupObject.firstName = result.fname;

    signupObject.lastName = result.lname;

    signupObject.userName = result.cname;

    signupObject.accountNumber = Number(result.accountNumber);

    signupObject.email = result.mailid;

    signupObject.phoneNumber = result.phoneno;

    signupObject.pasword = result.psw;

    this.service.postCustomerSignup(signupObject).subscribe((res) => {

      if (res) {

        alert('Sign Up Succesful, You can now login!');

      }

    });

  }

  send(): void {}

}

Deposit-main.css

.card {

    /\* Add shadows to create the "card" effect \*/

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    margin-top:5%;

    margin-right: 60%;

    margin-bottom: 2%;

    transition: 0.3s;

  }

  /\* On mouse-over, add a deeper shadow \*/

  .card:hover {

    box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

  }

  /\* Add some padding inside the card container \*/

  .container {

    padding: 16px 32px;

  }

  h4 {

    color: rgb(65, 65, 65);

  }

  .flex-container {

    display: flex;

    justify-content: space-between;

  }

  .float-card {

    position: absolute;

    top: 18%;

    right: 5%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  /\* .float-card:hover {

    box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

  } \*/

  .button {

    background-color: rgb(83, 83, 83);

    border: none;

    color: white;

    padding: 15px 32px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 16px;

  }

  .button:hover {

    background-color: black;

  }

  .float-card2 {

    position: absolute;

    top: 18%;

    right: 25%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  .float-button{

    position: absolute;

    top: 5%;

    right: 5%;

  }

  input[type="number"] {

    width: 25%;

    padding: 12px 20px;

    margin-left: 15%;

    margin-right: 60%;

    margin-bottom: 8px;

    display: inline-block;

    border: 1px solid #ccc;

    box-sizing: border-box;

  }

Deposit-money.html

<h2>Deposit Money</h2>

<div class="card">

    <div class="container">

        <div class="flex-container">

            <h4><b>Primary Balance: </b></h4>

            <h2><b>₹ {{data.primaryAmount}}</b></h2>

        </div>

    </div>

    <form #depositMoneyForm="ngForm" (ngSubmit)="depositMoney(depositMoneyForm.value)" >

        <div class="container" >

          <label for="deposit"><b>Amount: </b></label>

          <input type="number" placeholder="0.000" name="deposit" required ngModel>

          <input type="hidden" name="isDeposit" value="true">

          <input type="hidden" name="custId" value={{data.id}}>

          <button class="button">Deposit</button>

          </div>

    </form>

    <div class="float-button">

        <div class="container">

            <div class="flex-container">

                <button (click)="logoutHandler()" class="button">Log out</button>

            </div>

        </div>

    </div>

Deposit-money.specs.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { DepositMoneyComponent } from './deposit-money.component';

describe('DepositMoneyComponent', () => {

  let component: DepositMoneyComponent;

  let fixture: ComponentFixture<DepositMoneyComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ DepositMoneyComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(DepositMoneyComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

Deposit-Money.Ts

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

import { ActivatedRoute } from '@angular/router';

import { Router } from '@angular/router';

@Component({

  selector: 'app-deposit-money',

  templateUrl: './deposit-money.component.html',

  styleUrls: ['./deposit-money.component.css']

})

export class DepositMoneyComponent implements OnInit {

  public data: any=[];

  constructor(private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router) { }

  ngOnInit(): void {

    if(localStorage.getItem("CustomerId")){

      this.service.getCustomerDataById(parseInt(localStorage.getItem("CustomerId")!)).subscribe(res =>{

        this.data= res

      })

    }

    else{

      alert("Access forbidden you're being redirected to login page");

      this.router.navigate(["/login"])

    }

  }

  depositMoney(result:any[]):void{

    this.service.postDepositMoney(this.data.id,result).subscribe(res=>{

      if(res){

        alert("Money depositted successfully!");

        this.router.navigate(["/icinBank"]);

            }

      else{

        alert("Transaction declined, try after some time");

      }

    })

  }

  logoutHandler():void{

    localStorage.removeItem("CustomerId");

    this.router.navigate(["/login"]);

  }

}

LandingPage.css

.center{

    position: absolute;

    right:25%;

}

.welcome{

    font-size: large;

    font-family:'Courier New', Courier, monospace;

    font-weight: bolder ;

}

LadningPage.html

<p class="welcome">Welcome!</p>

<div class="center">

    <img [src]="myImage" alt="" width="800" height="500">

</div>

LandingPage.specs.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { LandingPageComponent } from './landing-page.component';

describe('LandingPageComponent', () => {

  let component: LandingPageComponent;

  let fixture: ComponentFixture<LandingPageComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ LandingPageComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(LandingPageComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

LandingPage.Ts

import { Component, OnInit } from '@angular/core';

@Component({

  selector: 'app-landing-page',

  templateUrl: './landing-page.component.html',

  styleUrls: ['./landing-page.component.css']

})

export class LandingPageComponent implements OnInit {

  title='bank';

  myImage: string = "assets/Images/bankImage.jpg"

  constructor() { }

  ngOnInit(): void {

  }

}

Transfer-Money.css

.card {

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    margin-top:5%;

    margin-right: 60%;

    margin-bottom: 2%;

    transition: 0.3s;

  }

  .card:hover {

    box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

  }

  .container {

    padding: 16px 32px;

  }

  h4 {

    color: rgb(65, 65, 65);

  }

  .flex-container {

    display: flex;

    justify-content: space-between;

  }

  .float-card {

    position: absolute;

    top: 18%;

    right: 5%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  .button {

    background-color: rgb(83, 83, 83);

    border: none;

    color: white;

    padding: 15px 32px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 16px;

  }

  .button:hover {

    background-color: black;

  }

  .float-card2 {

    position: absolute;

    top: 18%;

    right: 25%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  .float-button{

    position: absolute;

    top: 5%;

    right: 5%;

  }

  input[type="text"],

  input[type="number"] {

    width: 25%;

    padding: 12px 20px;

    margin-left: 15%;

    margin-right: 60%;

    margin-bottom: 8px;

    display: inline-block;

    border: 1px solid #ccc;

    box-sizing: border-box;

  }

Transfer-Money.html

<h2>Intra Bank Transfer</h2>

<div class="card">

    <div class="container">

        <div class="flex-container">

            <h4><b>Primary Balance: </b></h4>

            <h2><b>₹ {{data.primaryAmount}}</b></h2>

        </div>

    </div>

    <form #transferMoneyForm="ngForm" (ngSubmit)="transferMoney(transferMoneyForm.value)" >

        <div class="container" >

          <label for="beneficiaryName"><b>Benificiary Name: </b></label>

          <input type="text" placeholder="name" name="beneficiaryName" required ngModel>

          <label for="beneficiaryId"><b>Beneficiary Name: </b></label>

          <input type="number" placeholder="00000000" name="beneficiaryId" required ngModel>

          <label for="amount"><b>Amount: </b></label>

          <input type="number" placeholder="0.000" name="amount" required ngModel>

          <button class="button">Transfer</button>

          </div>

    </form>

    <div class="float-button">

        <div class="container">

            <div class="flex-container">

                <button (click)="logoutHandler()" class="button">Log out</button>

            </div>

        </div>

    </div>

Transfer-Money.specs.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { TranferMoneyComponent } from './tranfer-money.component';

describe('TranferMoneyComponent', () => {

  let component: TranferMoneyComponent;

  let fixture: ComponentFixture<TranferMoneyComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ TranferMoneyComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(TranferMoneyComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

TransferMoney.ts

import { Component, OnInit } from '@angular/core';

import { ActivatedRoute,Router } from '@angular/router';

import { ServiceService } from '../service.service';

@Component({

  selector: 'app-tranfer-money',

  templateUrl: './tranfer-money.component.html',

  styleUrls: ['./tranfer-money.component.css']

})

export class TranferMoneyComponent implements OnInit {

  public data: any=[];

  constructor(private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router) { }

  ngOnInit(): void {

    if(localStorage.getItem("CustomerId")){

      this.service.getCustomerDataById(parseInt(localStorage.getItem("CustomerId")!)).subscribe(res =>{

        this.data= res

      })

    }

    else{

      alert("Access forbidden you're being redirected to login page");

      this.router.navigate(["/login"])

    }

  }

  transferMoney(result:any[]):void{

    const sendData= {...result, customerId: this.data.id};

    this.service.postTransfer(sendData).subscribe((res)=>{

      if(res){

        alert("Money Transferred successfull!");

        this.router.navigate(["/icinBank"]);

            }

      else{

        alert("Transaction declined, try after some time");

      }

    })

  }

  logoutHandler():void{

    localStorage.removeItem("CustomerId");

    this.router.navigate(["/login"]);

  }

}

WithdrawMoney.css

.card {

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    margin-top:5%;

    margin-right: 60%;

    margin-bottom: 2%;

    transition: 0.3s;

  }

  .card:hover {

    box-shadow: 0 8px 16px 0 rgba(0, 0, 0, 0.2);

  }

  .container {

    padding: 16px 32px;

  }

  h4 {

    color: rgb(65, 65, 65);

  }

  .flex-container {

    display: flex;

    justify-content: space-between;

  }

  .float-card {

    position: absolute;

    top: 18%;

    right: 5%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  .button {

    background-color: rgb(83, 83, 83);

    border: none;

    color: white;

    padding: 15px 32px;

    text-align: center;

    text-decoration: none;

    display: inline-block;

    font-size: 16px;

  }

  .button:hover {

    background-color: black;

  }

  .float-card2 {

    position: absolute;

    top: 18%;

    right: 25%;

    box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);

    transition: 0.3s;

  }

  .float-button{

    position: absolute;

    top: 5%;

    right: 5%;

  }

  input[type="number"] {

    width: 25%;

    padding: 12px 20px;

    margin-left: 15%;

    margin-right: 60%;

    margin-bottom: 8px;

    display: inline-block;

    border: 1px solid #ccc;

    box-sizing: border-box;

  }

WithdrawMoney.html

<h2>Withdraw Money</h2>

<div class="card">

    <div class="container">

        <div class="flex-container">

            <h4><b>Primary Balance: </b></h4>

            <h2><b>₹ {{data.primaryAmount}}</b></h2>

        </div>

    </div>

    <form #withdrawMoneyForm="ngForm" (ngSubmit)="withdrawMoney(withdrawMoneyForm.value)" >

        <div class="container" >

          <label for="deposit"><b>Amount: </b></label>

          <input type="number" placeholder="0.000" name="deposit" required ngModel>

          <input type="hidden" name="isDeposit" value="true">

          <input type="hidden" name="custId" value={{data.id}}>

          <button class="button">Withdraw</button>

          </div>

    </form>

    <div class="float-button">

        <div class="container">

            <div class="flex-container">

                <button (click)="logoutHandler()" class="button">Log out</button>

            </div>

        </div>

    </div>

WithdrawMoney.specs.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';

import { WithdrawMoneyComponent } from './withdraw-money.component';

describe('WithdrawMoneyComponent', () => {

  let component: WithdrawMoneyComponent;

  let fixture: ComponentFixture<WithdrawMoneyComponent>;

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [ WithdrawMoneyComponent ]

    })

    .compileComponents();

    fixture = TestBed.createComponent(WithdrawMoneyComponent);

    component = fixture.componentInstance;

    fixture.detectChanges();

  });

  it('should create', () => {

    expect(component).toBeTruthy();

  });

});

WithdrawMoney.Ts

import { Component, OnInit } from '@angular/core';

import { ServiceService } from '../service.service';

import { ActivatedRoute, Router } from '@angular/router';

@Component({

  selector: 'app-withdraw-money',

  templateUrl: './withdraw-money.component.html',

  styleUrls: ['./withdraw-money.component.css']

})

export class WithdrawMoneyComponent implements OnInit {

  public data: any=[];

  constructor(private service: ServiceService,

    private route: ActivatedRoute,

    private router: Router) { }

  ngOnInit(): void {

    if(localStorage.getItem("CustomerId")){

      this.service.getCustomerDataById(parseInt(localStorage.getItem("CustomerId")!)).subscribe(res =>{

        this.data= res

      })

    }

    else{

      alert("Access forbidden you're being redirected to login page");

      this.router.navigate(["/login"])

    }

  }

  withdrawMoney(result:any[]):void{

    console.log(result);

    this.service.postWithdrawMoney(this.data.id,result).subscribe((res)=>{

      if(res){

        alert("Money withdrawal successfull!");

        this.router.navigate(["/icinBank"]);

            }

      else{

        alert("Transaction declined, try after some time");

      }

    })

  }

  logoutHandler():void{

    localStorage.removeItem("CustomerId");

    this.router.navigate(["/login"]);

  }

}

App-Routing

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

import { CommonModule } from '@angular/common';

import { AdminLoginComponent } from './admin-login/admin-login.component';

import { CustomerLoginComponent } from './customer-login/customer-login.component';

import { CustomerMainScreenComponent } from './customer-main-screen/customer-main-screen.component';

import { CustomerProfileComponent } from './customer-profile/customer-profile.component';

import { AdminScreenComponent } from './admin-screen/admin-screen.component';

import { CustomerRequestsComponent } from './customer-requests/customer-requests.component';

import { CustomerSignUpComponent } from './customer-sign-up/customer-sign-up.component';

import { ChequeBookRequestsComponent } from './cheque-book-requests/cheque-book-requests.component';

import { RouterModule, Routes } from '@angular/router';

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

import { LandingPageComponent } from './landing-page/landing-page.component';

import { WithdrawMoneyComponent } from './withdraw-money/withdraw-money.component';

import { TranferMoneyComponent } from './tranfer-money/tranfer-money.component';

import { DepositMoneyComponent } from './deposit-money/deposit-money.component';

const routes: Routes=[

  {

    path: 'Adminlogin', component: AdminLoginComponent

  },

  {

    path: 'login', component: CustomerLoginComponent

  },

  {

    path: 'icinBank', component: CustomerMainScreenComponent

  },

  {

    path: 'userProfile', component: CustomerProfileComponent

  },

  {

    path: 'Admin', component: AdminScreenComponent

  },

  {

    path: 'Admin/CustomerRequests', component: CustomerRequestsComponent

  },

  {

    path: 'icinBank/CustomerChequeBookRequest', component: ChequeBookRequestsComponent

  },

  {

    path: 'signup', component: CustomerSignUpComponent

  },

  {

    path: '', component: LandingPageComponent

  },

  {

    path:'icinBank/Withdraw', component: WithdrawMoneyComponent

  },

  {

    path:'icinBank/transfermoney', component: TranferMoneyComponent

  },

  {

    path:'icinBank/Deposit', component: DepositMoneyComponent

  }

]

@NgModule({

  imports: [RouterModule.forRoot(routes), HttpClientModule],

  exports: [RouterModule]

})

export class AppRoutingModule { }

AppComponent.css

ul {

  list-style-type: none;

  margin: 0;

  padding: 0;

  overflow: hidden;

  background-color: #333;

}

li {

  float: left;

}

li a {

  display: block;

  color: white;

  text-align: center;

  padding: 14px 16px;

  text-decoration: none;

}

/\* Change the link color to #111 (black) on hover \*/

li a:hover {

  background-color: #111;

}

.card {

  box-shadow: rgb(168, 168, 168);

  background-color: grey;

  margin-top: 5%;

  display: grid;

  width: 40%;

  height: 40%;

  grid-template-rows: max-content 200px 1fr;

}

.cards {

  display: grid;

  grid-template-columns: repeat(auto-fill, minmax(230px, 1fr));

  grid-gap: 20px;

}

App-Component.html

<ul>

  <li><a routerLink="/">Icin Bank</a></li>

  <li><a routerLink="/Adminlogin">Admin Login</a></li>

  <li><a routerLink="/login">Customer login</a></li>

  <li><a routerLink="/signup">Customer Sign-up</a></li>

</ul>

<router-outlet>

</router-outlet>

App-Component.specs.Ts

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

import { AppComponent } from './app.component';

describe('AppComponent', () => {

  beforeEach(async () => {

    await TestBed.configureTestingModule({

      declarations: [

        AppComponent

      ],

    }).compileComponents();

  });

  it('should create the app', () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app).toBeTruthy();

  });

  it(`should have as title 'icinbank'`, () => {

    const fixture = TestBed.createComponent(AppComponent);

    const app = fixture.componentInstance;

    expect(app.title).toEqual('icinbank');

  });

  it('should render title', () => {

    const fixture = TestBed.createComponent(AppComponent);

    fixture.detectChanges();

    const compiled = fixture.nativeElement as HTMLElement;

    expect(compiled.querySelector('.content span')?.textContent).toContain('icinbank app is running!');

  });

});

App-Component.Ts

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

@Component({

  selector: 'app-root',

  templateUrl: './app.component.html',

  styleUrls: ['./app.component.css']

})

export class AppComponent {

  title = 'icinbank';

}

App.Module.Ts

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

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

import { AppComponent } from './app.component';

import { AppRoutingModule } from './app-routing.module';

import { AdminLoginComponent } from './admin-login/admin-login.component';

import { CustomerLoginComponent } from './customer-login/customer-login.component';

import { CustomerMainScreenComponent } from './customer-main-screen/customer-main-screen.component';

import { CustomerProfileComponent } from './customer-profile/customer-profile.component';

import { AdminScreenComponent } from './admin-screen/admin-screen.component';

import { CustomerRequestsComponent } from './customer-requests/customer-requests.component';

import { CustomerSignUpComponent } from './customer-sign-up/customer-sign-up.component';

import { ChequeBookRequestsComponent } from './cheque-book-requests/cheque-book-requests.component';

import { FormsModule } from '@angular/forms';

import { CUSTOM\_ELEMENTS\_SCHEMA } from '@angular/core';

import { NoopAnimationsModule } from '@angular/platform-browser/animations';

import { MatCardModule } from '@angular/material/card';

import { LandingPageComponent } from './landing-page/landing-page.component';

import { TranferMoneyComponent } from './tranfer-money/tranfer-money.component';

import { DepositMoneyComponent } from './deposit-money/deposit-money.component';

import { WithdrawMoneyComponent } from './withdraw-money/withdraw-money.component';

@NgModule({

  declarations: [

    AppComponent,

    AdminLoginComponent,

    CustomerLoginComponent,

    CustomerMainScreenComponent,

    CustomerProfileComponent,

    AdminScreenComponent,

    CustomerRequestsComponent,

    CustomerSignUpComponent,

    ChequeBookRequestsComponent,

    LandingPageComponent,

    TranferMoneyComponent,

    DepositMoneyComponent,

    WithdrawMoneyComponent

  ],

  imports: [

    BrowserModule,

    AppRoutingModule,

    FormsModule,

    NoopAnimationsModule

  ],

  providers: [],

  bootstrap: [AppComponent]

})

export class AppModule { }

Service.service.spec.Ts

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

import { ServiceService } from './service.service';

describe('ServiceService', () => {

  let service: ServiceService;

  beforeEach(() => {

    TestBed.configureTestingModule({});

    service = TestBed.inject(ServiceService);

  });

  it('should be created', () => {

    expect(service).toBeTruthy();

  });

});

Service.service.Ts

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

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

@Injectable({

  providedIn: 'root'

})

export class ServiceService {

  constructor(private http: HttpClient) { }

  getTest(){

    return this.http.get<any>("http://localhost:8085/")

  }

  getSWAPI(){

    return this.http.get<any>("https://swapi.dev/api/people/1")

  }

  postCustomerSignup(obj:any){

    return this.http.post<any>("http://localhost:8085/customerSignUp",obj)

  }

  getLoginStatus(obj:any){

    return this.http.post<any>("http://localhost:8085/login",obj)

  }

  getAllUsersWithNoAccess(){

    return this.http.get<any>("http://localhost:8085/AllCustomersWithNoAccess")

  }

  getAllUsers(){

    return this.http.get<any>("http://localhost:8085/AllCustomers")

  }

  postBlock(obj:any){

    return this.http.post<any>("http://localhost:8085/blockCustomer",obj)

  }

  postCustomerLogin(obj:any){

    return this.http.post<any>("http://localhost:8085/customerLogin",obj);

  }

  getCustomerDataById(id:number){

    return this.http.get<any>(`http://localhost:8085/getCustomerById?id=${id}`);

  }

  postRequestChequeBook(id:number){

    return this.http.post<any>(`http://localhost:8085/requestChequeBook?id=${id}`,{

      "id": id

    });

  }

  postDepositMoney(id:number, dep:any){

    return this.http.post<any>("http://localhost:8085/transaction",{

      "customerId": id,

      "amount": dep.deposit,

      "isDeposit":true

    });

  }

  postWithdrawMoney(id:number, dep:any){

    return this.http.post<any>("http://localhost:8085/transaction",{

      "customerId": id,

      "amount": dep.deposit,

      "isDeposit":false

    });

  }

  postTransfer(obj:any){

    return this.http.post<any>("http://localhost:8085/Transfer",obj);

  }

  postAuthorizeCheque(customer:any){

    return this.http.post<any>("http://localhost:8085/AuthorizeCheque",customer);

  }

  postProvideAccess(customer:any){

    return this.http.post<any>("http://localhost:8085/AuthorizeAccess",customer);

  }

}