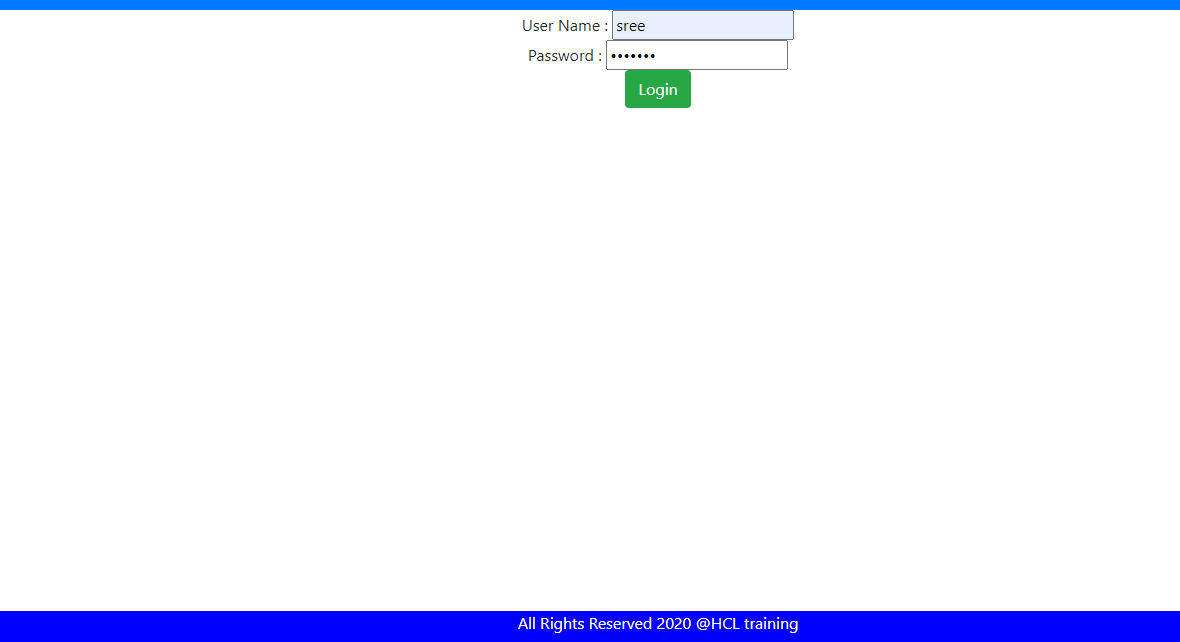
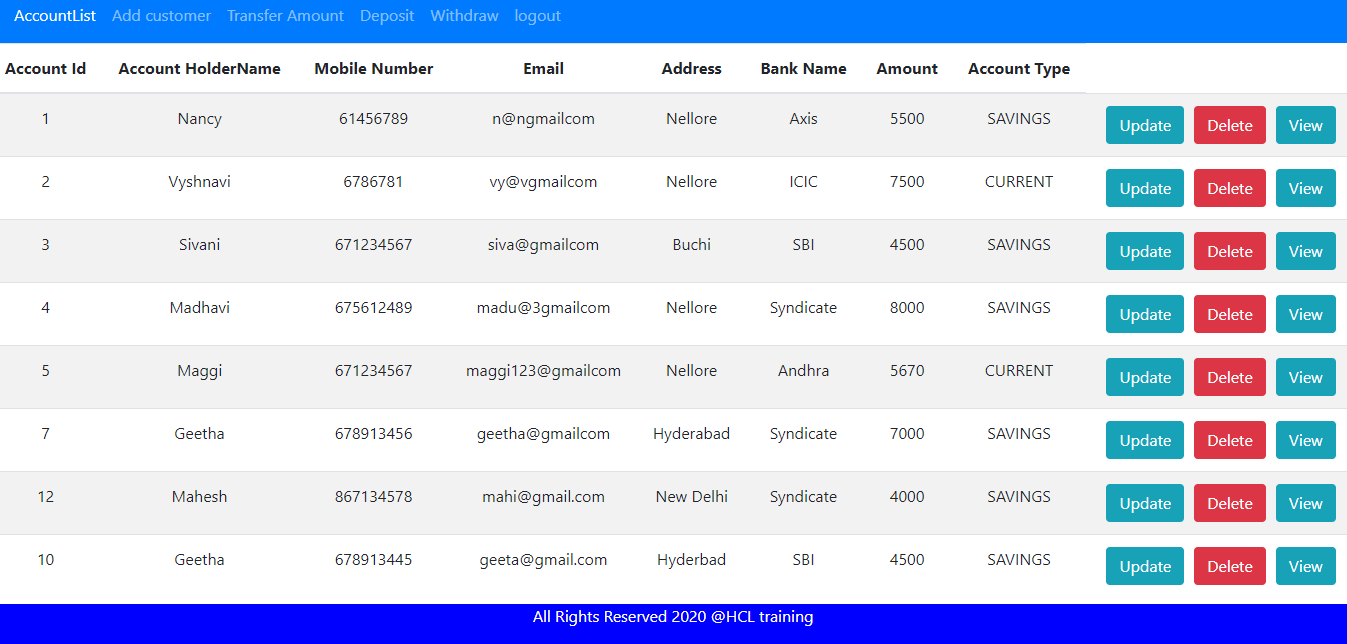
**Bank Application Document**

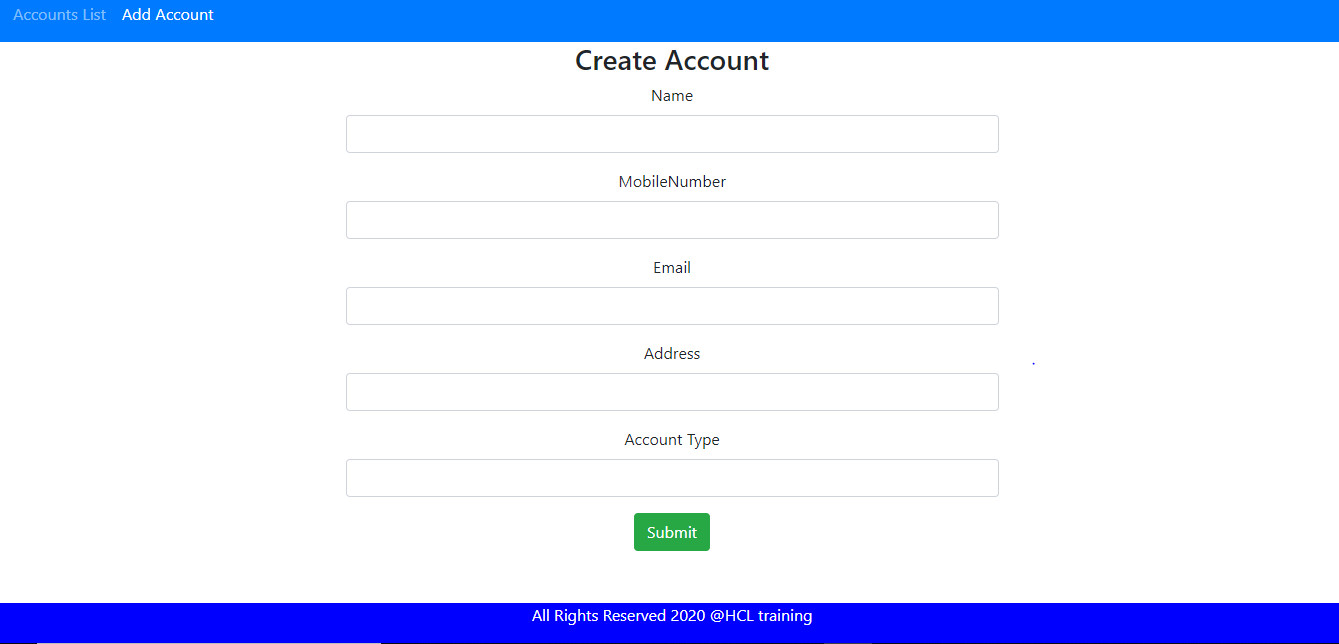
1. Log in to the page



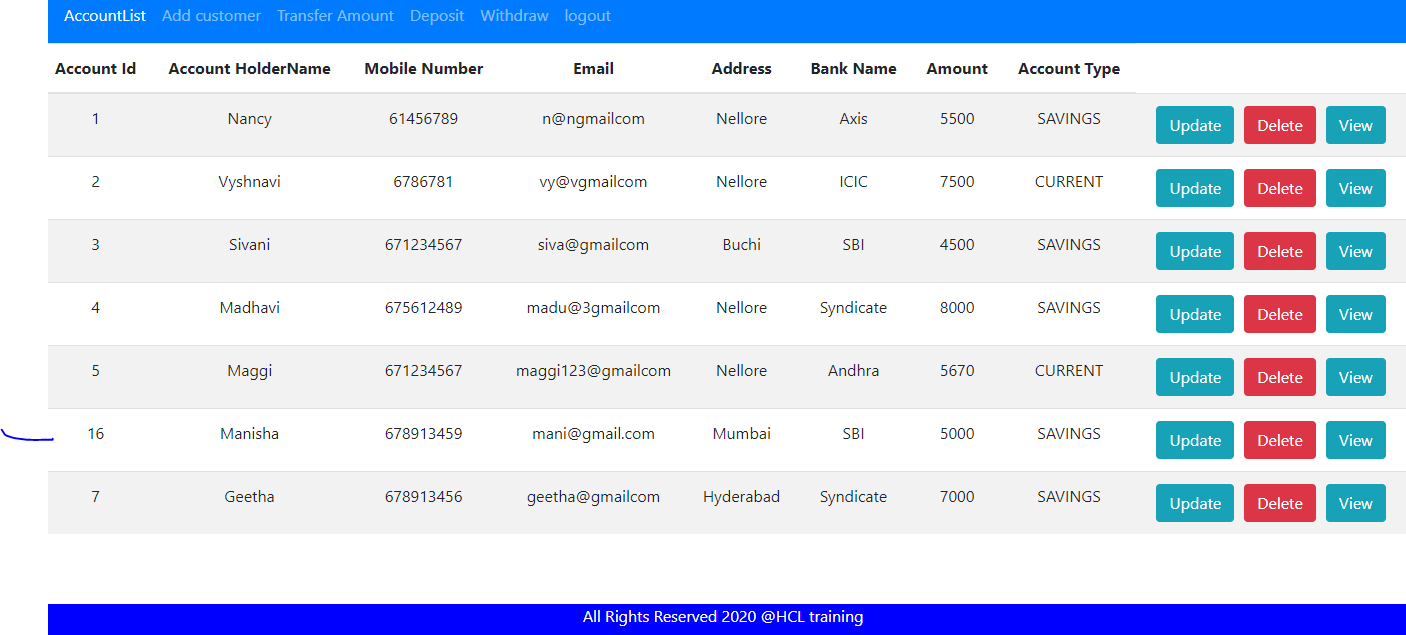
1. After Log In the total accounts list are given below



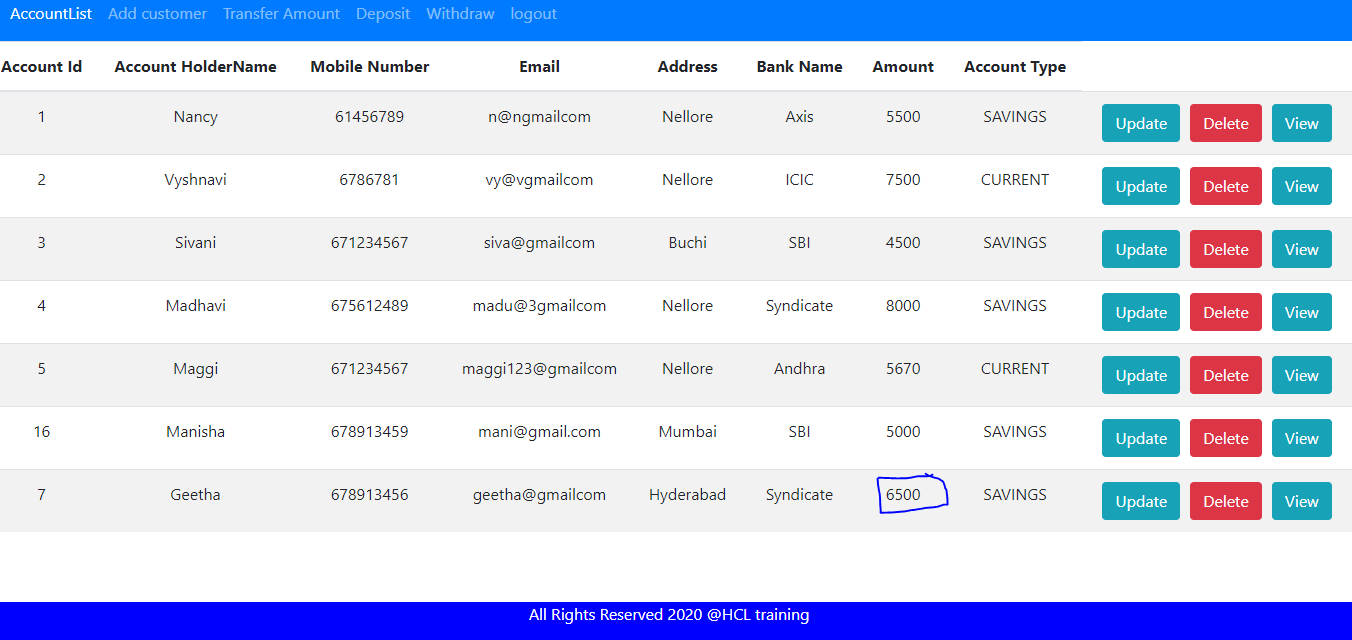
3. Creating a new Customer account



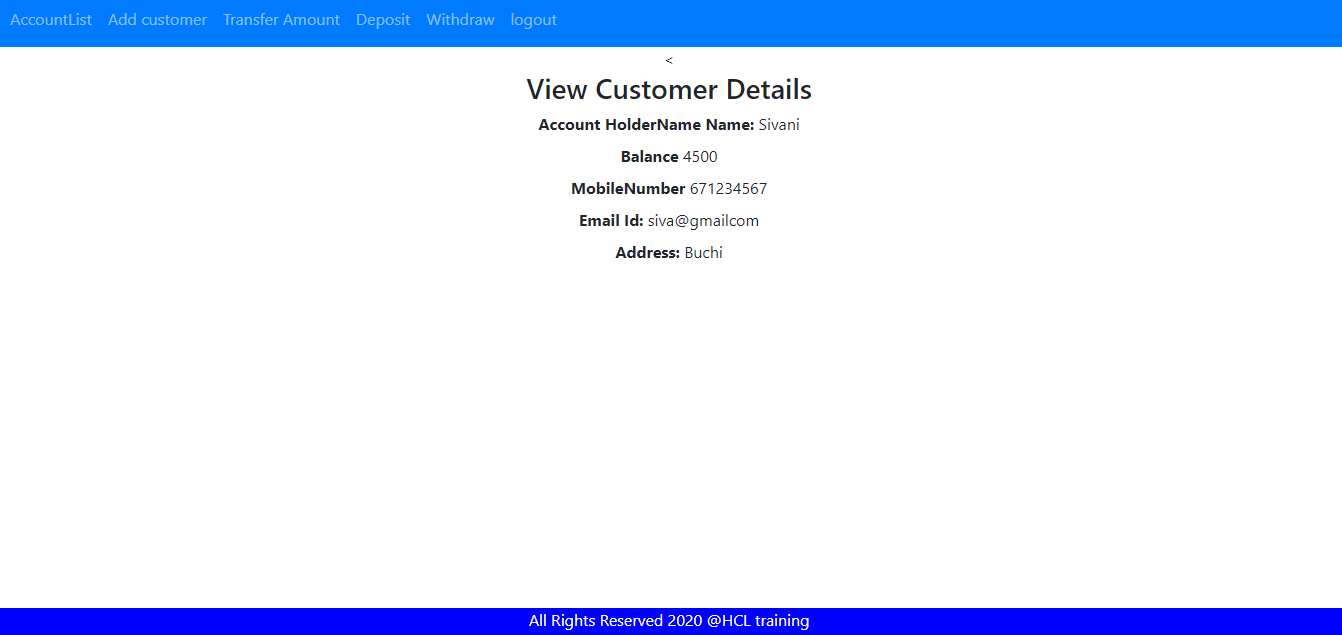
4.After adding new customer total accounts list are



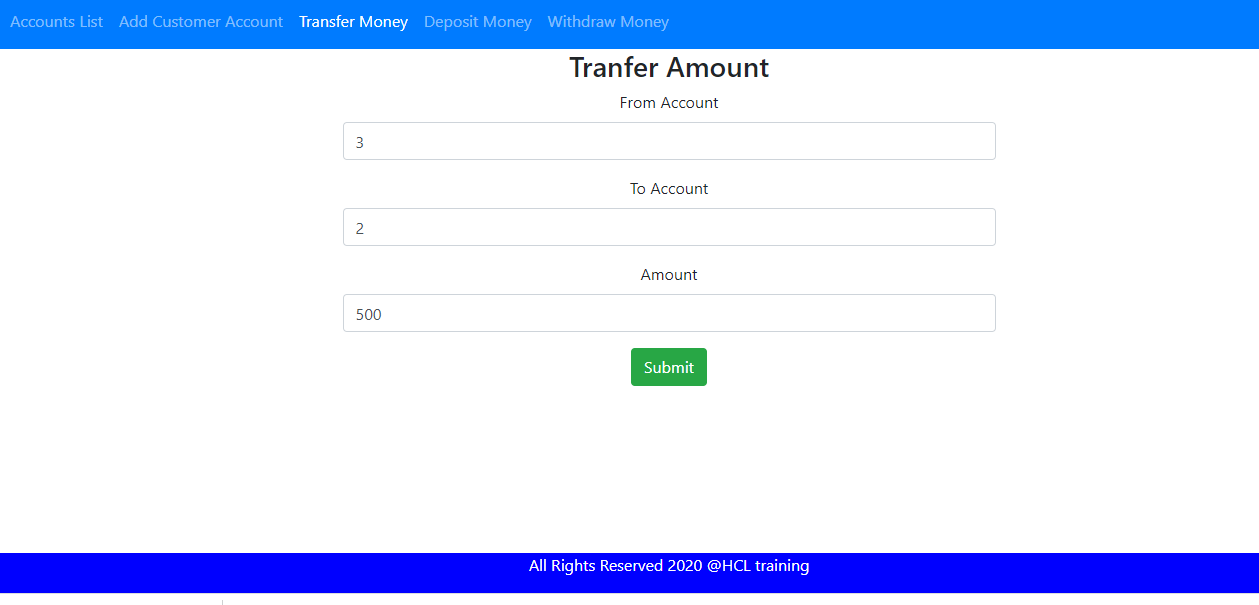
5. After updating the account details in Id:7



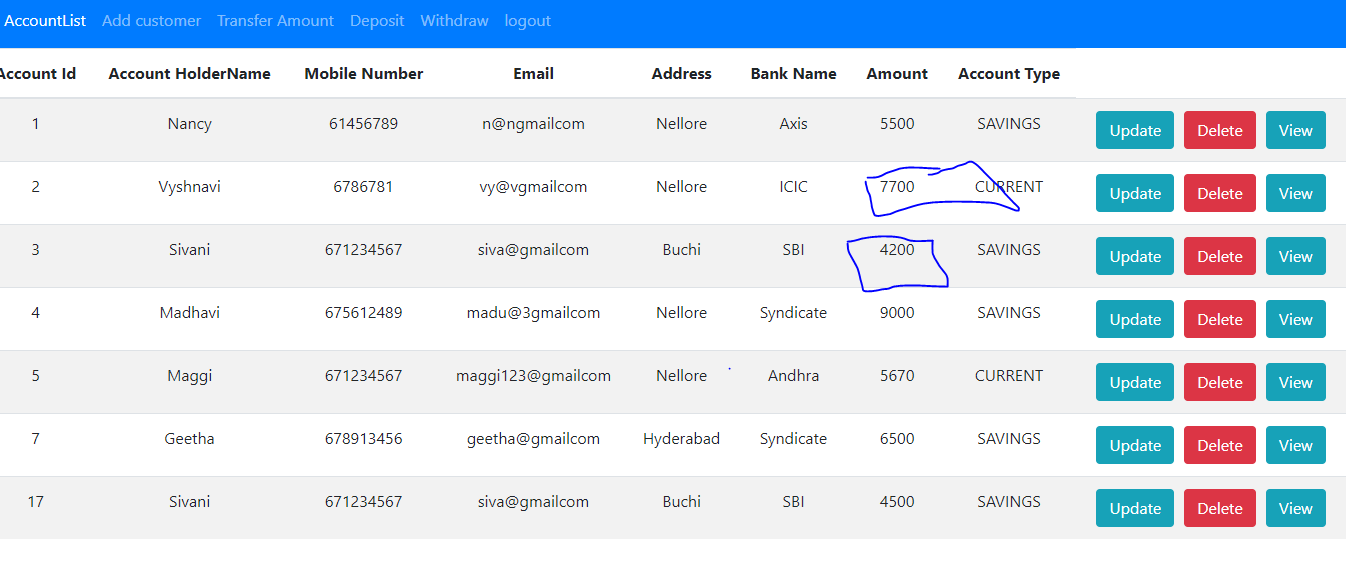
6. Viewing the account details by Id



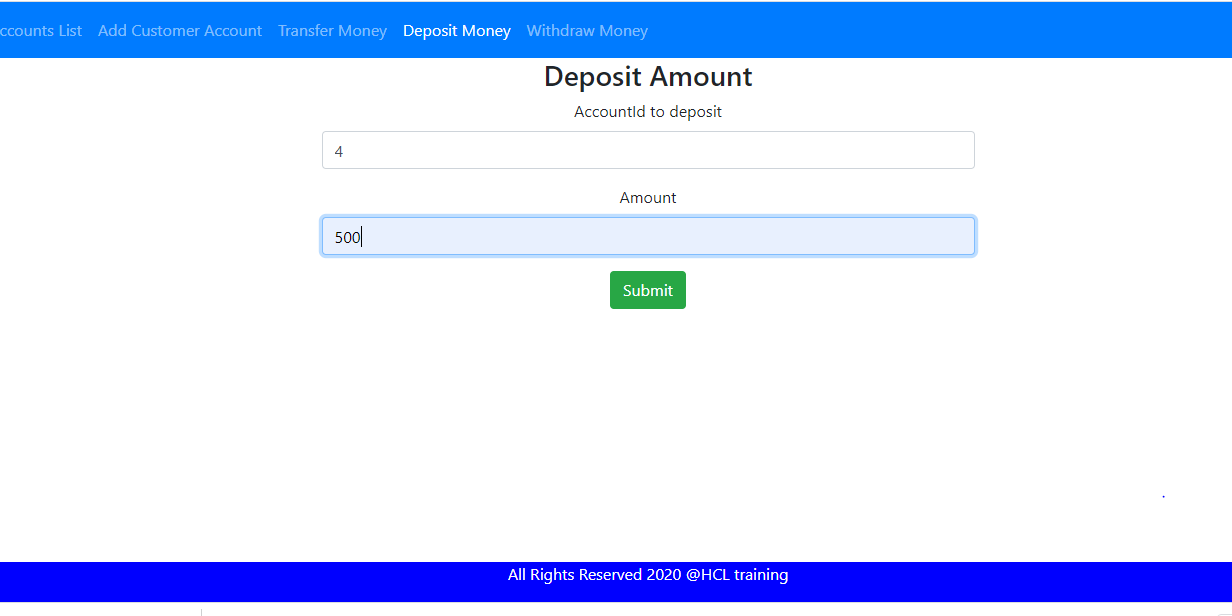
**7. Transferring the amount from One Id to another Id**

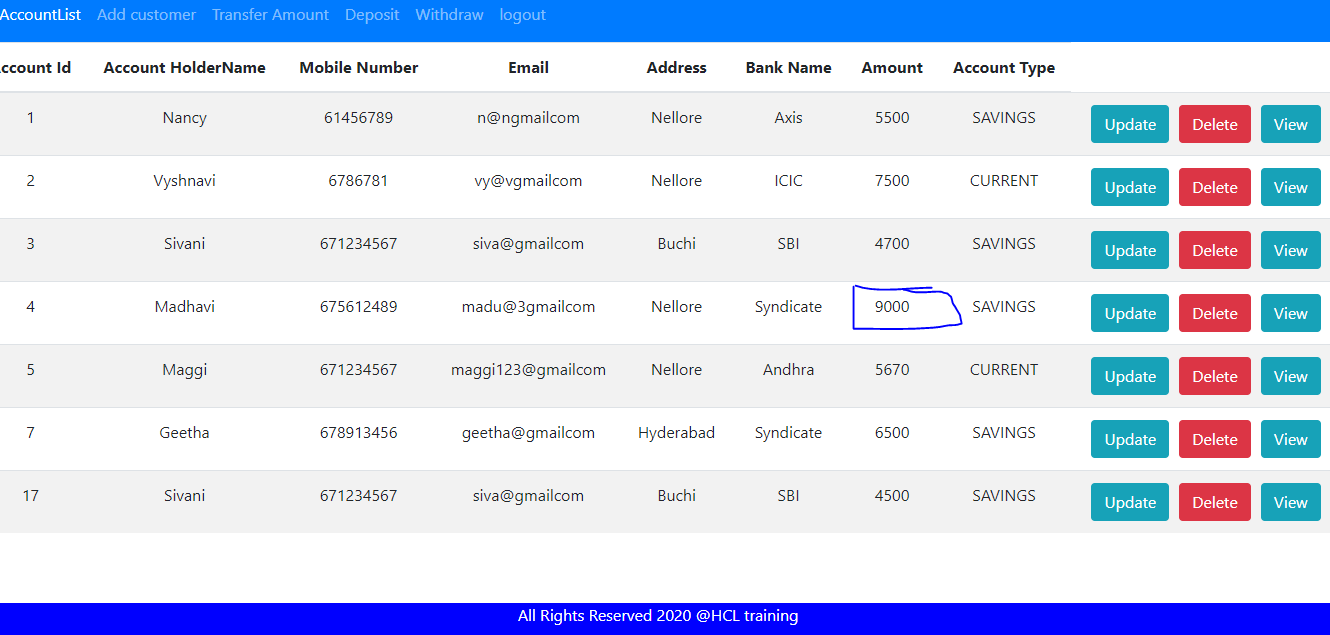


7. After transferring the amount the result was given below

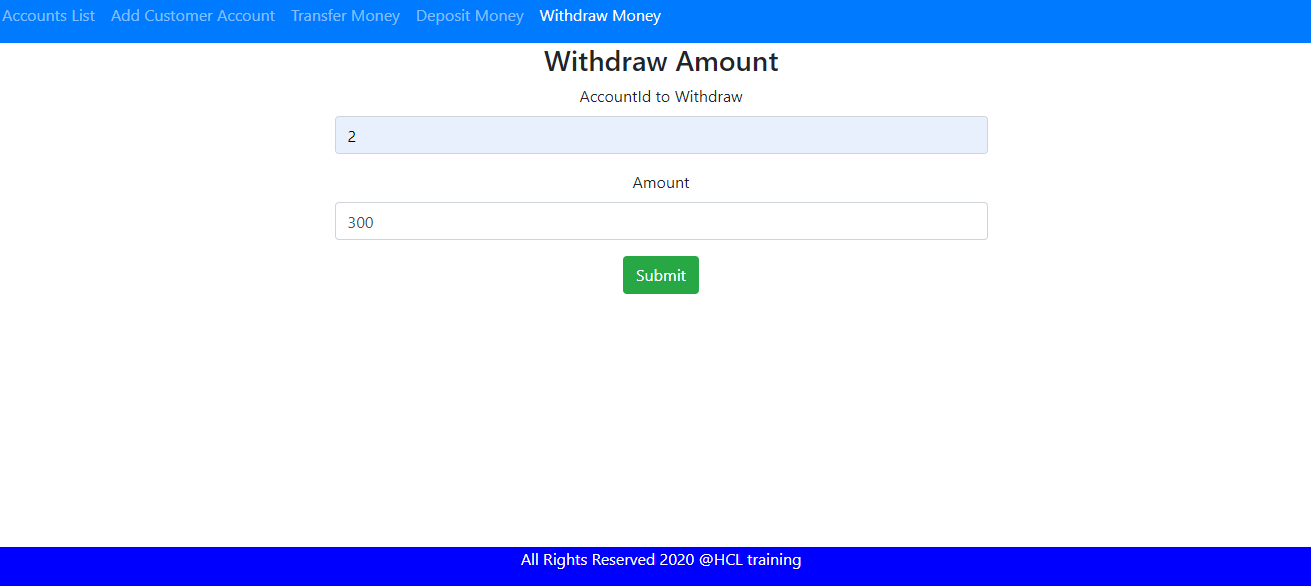


8. Depositing the amount using Id

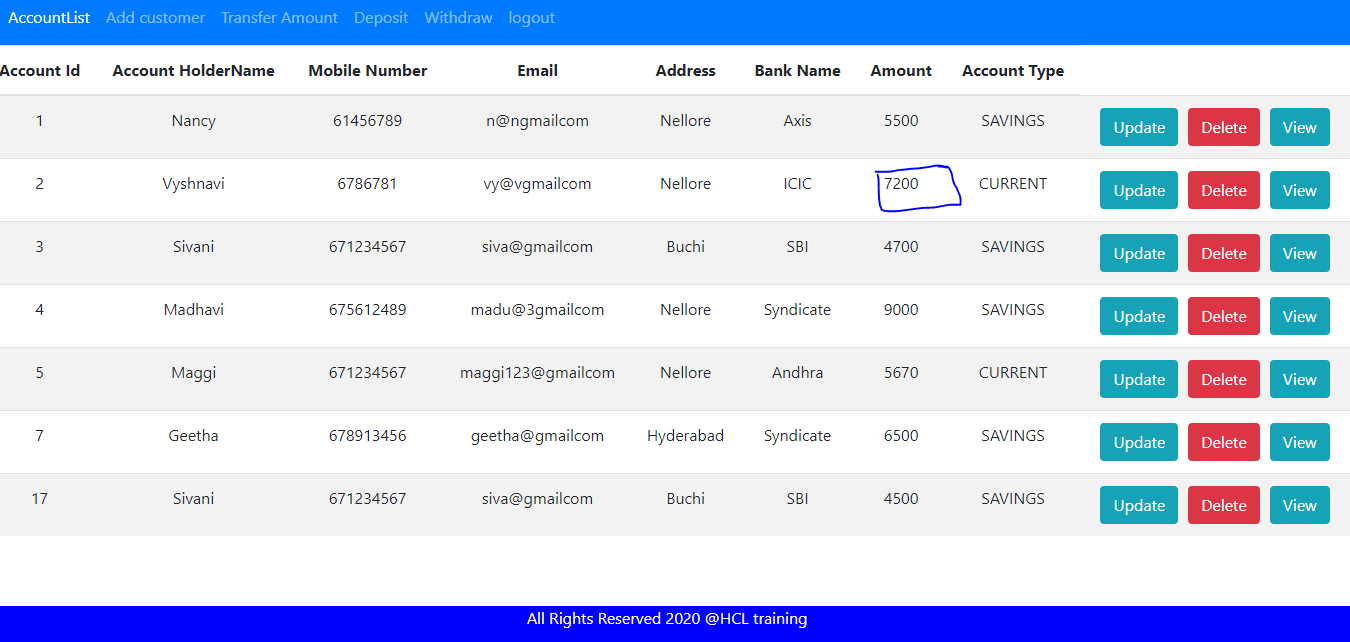
9. After depositing the amount using Id



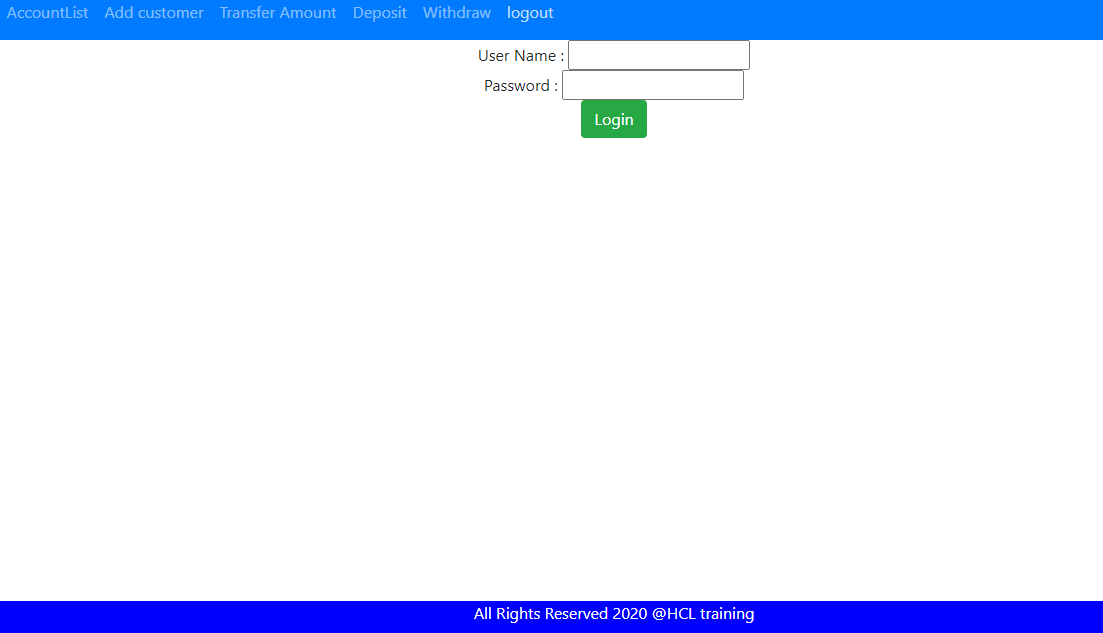
10. Withdrawing the money using Id



11. After withdrawing the amount using Id



12. After logout



**Code:**

**Account:**

**private** **int** accountId;

**private** String acoountHolderName;

**private** **int** mobileNum;

**private** String email;

**private** String address;

**private** String bankName;

**private** **int** amount;

**private** String accountType;

**Transaction:**

**private** **int** tId;

**private** **int** fromAccount;

**private** **int** toAccount;

@Temporal(TemporalType.***TIMESTAMP***)

**private** Date timeStamp;

**private** **int** tAmount;

**private** String message;

**private** String transType;

@ManyToOne(cascade=CascadeType.***ALL***,fetch=FetchType.***EAGER***)

@JoinColumn

**private** Account account

**Transaction Dao Layer**

import java.util.\*;

import com.bankapp.entities.TransactionLog;

public interface TransactionLogRepo {

public TransactionLog addCustomerTransactions(int fromAccountId,int toAccountId,Date timeStamp,String message,int tAmount,String txType);

public TransactionLog addCustomerTransactions(int fromAccountId,Date timeStamp,String message,int tAmount,String txType);

public TransactionLog findByTransactions(int accountId);

List<TransactionLog> getAllTransactionsDetails();

}

**Transaction Dao Implementation**

package com.bankapp.dao;

import java.util.Date;

import java.util.List;

import javax.persistence.EntityManager;

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

import org.springframework.stereotype.Repository;

import com.bankapp.entities.Account;

import com.bankapp.entities.TransactionLog;

@Repository

public class TransactionLogRepoImpl implements TransactionLogRepo{

@Autowired

private EntityManager em;

@Override

public TransactionLog addCustomerTransactions(int fromAccountId, int toAccountId, Date timeStamp, String message,

int tAmount, String txType) {

TransactionLog trasnsactions=new TransactionLog(fromAccountId, toAccountId, timeStamp, message, tAmount, txType);

em.persist(trasnsactions);

return trasnsactions;

}

@Override

public TransactionLog addCustomerTransactions(int fromAccountId, Date timeStamp, String message, int tAmount,

String txType) {

TransactionLog alltrasnsactions=new TransactionLog(fromAccountId, timeStamp, tAmount, message, txType);

em.persist(alltrasnsactions);

return alltrasnsactions;

}

@Override

public TransactionLog findByTransactions(int accountId) {

TransactionLog tansById=em.find(TransactionLog.class, accountId);

return tansById;

}

@Override

public List<TransactionLog> getAllTransactionsDetails() {

return em.createQuery("from TransactionLog").getResultList();

}

}

**AccountServive Implementation**

package com.bankapp.service;

import java.util.Date;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.bankapp.dao.AccountRepo;

import com.bankapp.entities.Account;

import com.bankapp.exceptions.AccountNotFound;

@Service

@Transactional

public class AccountServiceImpl implements AccountService{

@Autowired

private AccountRepo accountRepo;

@Autowired

private TransactionLogService transService;

@Override

public List<Account> getAllCustomerAccounts() {

return accountRepo.findAll();

}

@Override

public Account addCustomerAccount(Account account) {

accountRepo.save(account);

return account;

}

@Override

public Account updateCustomerAccount(int accountId, Account account) {

Account accountToUpdate=getCustomerAccountById(accountId);

accountToUpdate.setAddress(account.getAddress());

accountToUpdate.setEmail(account.getEmail());

accountToUpdate.setAmount(account.getAmount());

accountRepo.save(accountToUpdate);

return accountToUpdate;

}

@Override

public Account deleteCustomerAccount(int accountId) {

Account accountToDelete=getCustomerAccountById(accountId);

accountRepo.delete(accountToDelete);

return accountToDelete;

}

@Override

public Account getCustomerAccountById(int accountId) {

return accountRepo.findById(accountId).orElseThrow(()->new AccountNotFound("Account details not found"));

}

@Override

public String tranferMoney(int fromAccountId, int toAccountId, int amount) {

Account accountFrom=withdrawMoney(fromAccountId,amount);

Account accountTo=depositMoney(toAccountId,amount);

accountRepo.save(accountFrom);

accountRepo.save(accountTo);

transService.addCustomerTransactions(fromAccountId, toAccountId, new Date(), "Success", amount,"debited");

return "transferred";

}

@Override

public Account depositMoney(int accountId, int amount) {

Account amountToDeposit=getCustomerAccountById(accountId);

amountToDeposit.setAmount(amountToDeposit.getAmount()+amount);

accountRepo.save(amountToDeposit);

transService.addCustomerTransactions(accountId, new Date(), "Deposited",amount, "credited");

return amountToDeposit;

}

@Override

public Account withdrawMoney(int accountId, int amount) {

Account amountToWithdraw=getCustomerAccountById(accountId);

amountToWithdraw.setAmount(amountToWithdraw.getAmount()-amount);

accountRepo.save(amountToWithdraw);

transService.addCustomerTransactions(accountId, new Date(), "withdrawn", amount, "debited");

return amountToWithdraw;

}

}

**Transactional Service Implementation**

package com.bankapp.service;

import java.util.Date;

import java.util.List;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.bankapp.dao.TransactionLogRepo;

import com.bankapp.entities.TransactionLog;

@Service

@Transactional

public class TransactionLogServiceImpl implements TransactionLogService{

@Autowired

private TransactionLogRepo transRepo;

public TransactionLog addCustomerTransactions(int fromAccountId,int toAccountId,Date timeStamp,String message,int tAmount,String txType) {

return transRepo.addCustomerTransactions(fromAccountId, toAccountId, timeStamp, message, tAmount, txType);

}

@Override

public TransactionLog addCustomerTransactions(int fromAccountId, Date timeStamp, String message, int tAmount,

String txType) {

return transRepo.addCustomerTransactions(fromAccountId, timeStamp, message, tAmount, txType);

}

@Override

public TransactionLog findByTransactions(int accountId) {

return transRepo.findByTransactions(accountId);

}

@Override

public List<TransactionLog> getAllTransactionsDetails() {

return transRepo.getAllTransactionsDetails();

}

}

**Controller**

package com.bankapp.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.MediaType;

import org.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

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

import com.bankapp.beans.DepositBeans;

import com.bankapp.beans.TransferBeans;

import com.bankapp.beans.WithdrawBeans;

import com.bankapp.entities.Account;

import com.bankapp.entities.TransactionLog;

import com.bankapp.exceptions.TransactionDetailsNotFound;

import com.bankapp.secconfig.AuthResponse;

import com.bankapp.service.AccountService;

import com.bankapp.service.TransactionLogService;

import java.util.\*;

@RestController

@CrossOrigin(origins = "\*")

public class BankManagerController {

@Autowired

private AccountService accountService;

@Autowired

private TransactionLogService transService;

@GetMapping(path="/accounts",produces=MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<List<Account>> getCustomerAccount(){

List<Account> allAccounts=accountService.getAllCustomerAccounts();

return new ResponseEntity<List<Account>>(allAccounts,HttpStatus.OK);

}

@GetMapping(path="/accounts/{id}",produces=MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<Account> getCustomerAccountById(@PathVariable(name="id") int accountId){

Account accountById=accountService.getCustomerAccountById(accountId);

return new ResponseEntity<Account>(accountById,HttpStatus.OK);

}

@PostMapping(path="/accounts",produces=MediaType.APPLICATION\_JSON\_VALUE,consumes=MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<Account> addNewCustomerAccount(@RequestBody Account account){

Account accountAdding=accountService.addCustomerAccount(account);

return new ResponseEntity<Account>(accountAdding,HttpStatus.CREATED);

}

@PutMapping(path="/accounts/{id}",produces=MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<Account> getUpdatedCustomerAccount(@PathVariable(name="id") int accountId,@RequestBody Account account){

Account accountToUpdated=accountService.updateCustomerAccount(accountId, account);

return new ResponseEntity<Account>(accountToUpdated,HttpStatus.CREATED);

}

@DeleteMapping(path="/accounts/{id}",produces=MediaType.APPLICATION\_JSON\_VALUE)

public ResponseEntity<Void> getCustomerAccountDeleted(@PathVariable(name="id") int accountId){

Account accountToDeleted=accountService.deleteCustomerAccount(accountId);

return new ResponseEntity<Void>(HttpStatus.NO\_CONTENT);

}

@PostMapping(path="/accounts/transfer",produces=MediaType.APPLICATION\_JSON\_VALUE)

public String getTransferDetails(@RequestBody TransferBeans transferBeans) {

String accountTransfer=accountService.tranferMoney(transferBeans.getFromAccountId(), transferBeans.getToAccountId(),transferBeans.getAmount());

if(accountTransfer==null)

return "Not transferred";

else

return "successfully transferred";

}

@PostMapping(path="/accounts/deposit",produces=MediaType.APPLICATION\_JSON\_VALUE,consumes=MediaType.APPLICATION\_JSON\_VALUE)

public Account getDepositDetails(@RequestBody DepositBeans depositBeans) {

Account accountToBeDeposited=accountService.depositMoney(depositBeans.getDepositaccountId(), depositBeans.getAmountToDeposit());

return accountToBeDeposited;

}

@PostMapping(path="/accounts/withdraw",produces=MediaType.APPLICATION\_JSON\_VALUE,consumes=MediaType.APPLICATION\_JSON\_VALUE)

public Account getWithdrawDetails(@RequestBody WithdrawBeans withdrawBeans) {

Account accountToBeWithdrawn=accountService.withdrawMoney(withdrawBeans.getWithdrawAccountId(), withdrawBeans.getAmountToWithdraw());

return accountToBeWithdrawn;

}

@GetMapping(path="/accounts/transaction",produces=MediaType.APPLICATION\_JSON\_VALUE)

public List<TransactionLog> getAlTransaction() {

List <TransactionLog> allTrans=transService.getAllTransactionsDetails();

return allTrans;

}

@GetMapping(path="/accounts/transaction/{id}",produces=MediaType.APPLICATION\_JSON\_VALUE)

public TransactionLog getTransaction(@PathVariable (name="id")int id) {

TransactionLog accountTrans=transService.findByTransactions(id);

return accountTrans;

}

@GetMapping(produces = "application/json")

@RequestMapping({ "/validateLogin" })

public AuthResponse validateLogin() {

return new AuthResponse("User successfully authenticated");

}

}

**Beans:**

**Transfer:**

**private** **int** fromAccountId;

**private** **int** toAccountId;

**private** **int** amount;

**Deposit:**

**private** **int** depositaccountId;

**private** **int** amountToDeposit;

**Withdraw:**

**private** **int** withdrawAccountId;

**private** **int** amountToWithdraw;

**Security Configuration**

package com.bankapp.secconfig;

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

import org.springframework.http.HttpMethod;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter{

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable().

authorizeRequests().antMatchers(HttpMethod.OPTIONS, "/\*\*").permitAll().anyRequest().authenticated()

.and().httpBasic();

}

@Autowired

public void configureGlobal(AuthenticationManagerBuilder auth) throws Exception {

auth.inMemoryAuthentication().withUser("sree").password("{noop}sree123").roles("USER");

}

}

**Angular Code:**

**AccountDetails Class:**

accountId:number;

  acoountHolderName:String

     mobileNum:number;

     email:String;

    address:String;

    bankName:String;

    amount:number;

    accountType:String;

**AccountComponent:**

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

import {AccountDetails} from '../accountdetails';

import { AccountService } from '../account.service';

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

@Component({

  selector: 'app-account',

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

  styleUrls: ['./account.component.css']

})

export class AccountComponent implements OnInit {

  accounts: AccountDetails[];

  constructor(private accountService: AccountService,private router:Router) { }

  ngOnInit() {

    this.getAccounts();

  }

  private getAccounts(){

    this.accountService.getAccountList().subscribe(

      data=>{this.accounts=data});

  }

  updateAccount(id: number){

    console.log(`-----------`)

    this.router.navigate(['update-account', id]);

  }

  deleteAccount(accountId: number){

    this.accountService.deleteAccount(accountId).subscribe(data=>{

      this.getAccounts();

      console.log(data);

    })

  }

  customerDetails(accountId: number){

    this.router.navigate(['customeraccount-details', accountId]);

  }

}

**Account service:**

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

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

import { from, Observable } from 'rxjs';

import { AccountDetails } from './accountdetails';

import { Deposit } from './deposit';

import { Transfer } from './transfer';

import { Withdraw } from './withdraw';

@Injectable({

  providedIn: 'root'

})

export class AccountService {

  private baseURL="http://localhost:8089/bankapp/accounts";

  constructor(private httpClient: HttpClient) { }

//return this.httpClient.get<Employee[]>(`${this.baseURL}`);

  getAccountList(): Observable<AccountDetails[]>{

    return this.httpClient.get<AccountDetails[]>(`${this.baseURL}`);

  }

  createAccount(account: AccountDetails): Observable<Object>{

    return this.httpClient.post(`${this.baseURL}`, account);

  }

  updateEmployee(accountId: number, account: AccountDetails): Observable<Object>{

    return this.httpClient.put(`${this.baseURL}/${accountId}`, account);

  }

  getAccountById(accountId: number): Observable<AccountDetails>{

    return this.httpClient.get<AccountDetails>(`${this.baseURL}/${accountId}`);

  }

  deleteAccount(accountId: number): Observable<Object>{

    return this.httpClient.delete(`${this.baseURL}/${accountId}`);

  }

   depositAmount(deposit: Deposit): Observable<Object>{

     return this.httpClient.put(`${this.baseURL}`,deposit);

   }

 withdrawAmount(withdraw: Withdraw): Observable<Object>{

  return this.httpClient.put(`${this.baseURL}`,withdraw);

 }

 transferAmount(transfers: Transfer): Observable<Object>{

  return this.httpClient.put(`${this.baseURL}`,transfers);

 }

}

**Create-Account:**

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

import { AccountDetails } from '../accountdetails';

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

import { AccountService } from '../account.service';

@Component({

  selector: 'app-create-account',

  templateUrl: './create-account.component.html',

  styleUrls: ['./create-account.component.css']

})

export class CreateAccountComponent implements OnInit {

  account: AccountDetails = new AccountDetails();

  constructor(private accountService: AccountService, private router: Router) { }

  ngOnInit(): void {

  }

  saveAccount(){

    this.accountService.createAccount(this.account).subscribe( data =>{

      console.log(data);

      this.goToAccountList();

    },

    error => console.log(error));

  }

  goToAccountList(){

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

  }

  onSubmit(){

    console.log(this.account);

    this.saveAccount();

  }

}

**HTML Page:**

<div class="col-md-6 offset-md-3">

    <h3> Create Account </h3>

    <form (ngSubmit) = "onSubmit()">

        <div class="form-group">

            <label> Name</label>

            <input type="text" class ="form-control" id = "name"

                [(ngModel)] = "account.acoountHolderName" name = "name">

        </div>

        <div class="form-group">

            <label> MobileNumber </label>

            <input type="text" class ="form-control" id = "mobilenum"

                [(ngModel)] = "account.mobileNum" name = "mobile">

        </div>

        <div class="form-group">

            <label> Email </label>

            <input type="text" class ="form-control" id = "email"

                [(ngModel)] = "account.email" name = "email">

        </div>

              <div class="form-group">

            <label> Address </label>

            <input type="text" class ="form-control" id = "address"

                [(ngModel)] = "account.address" name = "address">

        </div>

              <div class="form-group">

            <label> Bank Name </label>

            <input type="text" class ="form-control" id = "bankname"

                [(ngModel)] = "account.bankName" name = "bankname">

        </div>

        <div class="form-group">

            <label> Amount </label>

            <input type="text" class ="form-control" id = "amount"

                [(ngModel)] = "account.amount" name = "amount">

        </div>

        <div class="form-group">

            <label> Account Type </label>

            <input type="text" class ="form-control" id = "accounttype"

                [(ngModel)] = "account.accountType" name = "accounttype">

        </div>

        <button class = "btn btn-success" type ="submit">Submit</button>

    </form>

    </div>

**Update-Account:**

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

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

import { AccountService } from '../account.service';

import { AccountDetails } from '../accountdetails';

@Component({

  selector: 'app-update-account',

  templateUrl: './update-account.component.html',

  styleUrls: ['./update-account.component.css']

})

export class UpdateAccountComponent implements OnInit {

  accountId: number;

  account: AccountDetails = new AccountDetails();

  constructor(private accountService: AccountService,

    private route: ActivatedRoute, private router: Router) { }

  ngOnInit(): void {

    this.accountId=this.route.snapshot.params['accountId'];

   this.accountService.getAccountById(this.accountId).subscribe(data=>{

    this.account=data;

    }, error=>console.log(error))

  }

  onSubmit(){

    this.accountService.updateEmployee(this.accountId, this.account)

    .subscribe(data=> {

        this.goToAccountList();

    }, error=> console.log(error))

  }

goToAccountList(){

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

  }

}

**Html page:**

<div class="col-md-6 offset-md-3">

    <h3> Create Account </h3>

    <form (ngSubmit) = "onSubmit()">

        <div class="form-group">

            <label> Name</label>

            <input type="text" class ="form-control" id = "name"

                [(ngModel)] = "account.acoountHolderName" name = "name">

        </div>

        <div class="form-group">

            <label> MobileNumber </label>

            <input type="text" class ="form-control" id = "mobilenum"

                [(ngModel)] = "account.mobileNum" name = "mobile">

        </div>

        <div class="form-group">

            <label> Email </label>

            <input type="text" class ="form-control" id = "email"

                [(ngModel)] = "account.email" name = "email">

        </div>

              <div class="form-group">

            <label> Address </label>

            <input type="text" class ="form-control" id = "address"

                [(ngModel)] = "account.address" name = "address">

        </div>

              <div class="form-group">

            <label> Bank Name </label>

            <input type="text" class ="form-control" id = "bankname"

                [(ngModel)] = "account.bankName" name = "bankname">

        </div>

        <div class="form-group">

            <label> Amount </label>

            <input type="text" class ="form-control" id = "amount"

                [(ngModel)] = "account.amount" name = "amount">

        </div>

        <div class="form-group">

            <label> Account Type </label>

            <input type="text" class ="form-control" id = "accounttype"

                [(ngModel)] = "account.accountType" name = "accounttype">

        </div>

        <button class = "btn btn-success" type ="submit">Submit</button>

    </form>

    </div>

**AccountDetails:**

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

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

import { AccountService } from '../account.service';

import { AccountDetails } from '../accountdetails';

@Component({

  selector: 'app-customeraccount-details',

  templateUrl: './customeraccount-details.component.html',

  styleUrls: ['./customeraccount-details.component.css']

})

export class CustomeraccountDetailsComponent implements OnInit {

accountId: number;

account: AccountDetails=new AccountDetails();

  constructor(private accountService: AccountService,

    private router: ActivatedRoute) { }

    ngOnInit(): void {

      this.accountId = this.router.snapshot.params['accountId'];

      this.account=new AccountDetails();

  this.accountService.getAccountById(this.accountId).subscribe(data=>{

    this.account=data;

  });

}

}

**AppComponent.HTML**

<nav class="navbar navbar-expand-sm bg-primary navbar-dark">

  <ul class = "navbar-nav">

    <li class = "nav-item">

      <a [routerLink]="['/login']" routerLinkActive="active" class="nav-link" >login</a>

  </li>

      <li class = "nav-item">

          <a [routerLink]="['/accounts']" routerLinkActive="active" class="nav-link" >AccountList</a>

      </li>

      <li class = "nav-item">

        <a  [routerLink]="['/create-account']" routerLinkActive="active" class="nav-link" >Add customer</a>

      </li>

      <li class = "nav-item">

        <a  [routerLink]="['/transfer']" routerLinkActive="active" class="nav-link" >Transfer Amount</a>

      </li>

      <li class = "nav-item">

        <a  [routerLink]="['/deposit']" routerLinkActive="active" class="nav-link" >Deposit</a>

      </li>

      <li class = "nav-item">

        <a  [routerLink]="['/withdraw']" routerLinkActive="active" class="nav-link" >Withdraw</a>

      </li>

      <li class = "nav-item">

        <a   [routerLink]="['/logout']" routerLinkActive="active" class="nav-link" >logout</a>

      </li>

  </ul>

</nav>

<div class="text-center">

  <router-outlet></router-outlet>

</div>

<footer class = "footer">

  <div class = "container">

      <span>All Rights Reserved 2020 @HCL training</span>

  </div>

</footer>

**App routing Module**

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

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

import { AccountComponent } from './account/account.component';

import { AuthGaurdService } from './auth-gaurd.service';

import { CreateAccountComponent } from './create-account/create-account.component';

import { CustomeraccountDetailsComponent } from './customeraccount-details/customeraccount-details.component';

import { DepositComponent } from './deposit/deposit.component';

import { LoginComponent } from './login/login.component';

import { LogoutComponent } from './logout/logout.component';

import { TransferComponent } from './transfer/transfer.component';

import { UpdateAccountComponent } from './update-account/update-account.component';

import { WithdrawComponent } from './withdraw/withdraw.component';

const routes: Routes = [

  {path:'accounts', component: AccountComponent,canActivate:[AuthGaurdService]},

  {path:'create-account', component: CreateAccountComponent,canActivate:[AuthGaurdService]},

  {path:'update-account/:accountId', component: UpdateAccountComponent,canActivate:[AuthGaurdService]},

  {path:'customeraccount-details/:accountId', component: CustomeraccountDetailsComponent,canActivate:[AuthGaurdService]},

  {path:'transfer', component: TransferComponent,canActivate:[AuthGaurdService]},

  {path:'deposit', component: DepositComponent,canActivate:[AuthGaurdService]},

  {path:'withdraw', component: WithdrawComponent,canActivate:[AuthGaurdService]},

  { path: 'login', component: LoginComponent },

  { path: 'logout', component: LogoutComponent ,canActivate:[AuthGaurdService]},

  {path:'', redirectTo:'accounts', pathMatch:'full'}

];

@NgModule({

  imports: [RouterModule.forRoot(routes)],

  exports: [RouterModule]

})

export class AppRoutingModule { }

**Authentication:**

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

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

import { map } from 'rxjs/operators';

export class AuthResponse{

  constructor(public status:string) {}

}

@Injectable({

  providedIn: 'root'

})

export class AuthenticationService {

  constructor(private httpClient:HttpClient) {  }

  authenticate(username, password) {

    const headers = new HttpHeaders({ Authorization: 'Basic ' + btoa(username + ':' + password) });

    return this.httpClient.get<AuthResponse>('http://localhost:8089/bankapp/validateLogin',{headers}).pipe(

      map(

        userData => {

          sessionStorage.setItem('username', username);

          let authString = 'Basic ' + btoa(username + ':' + password);

          sessionStorage.setItem('basicauth', authString);

          return userData;

        }

      )

    );

  }

  isUserLoggedIn() {

    let user = sessionStorage.getItem('username')

    console.log(!(user === null))

    return !(user === null)

  }

  logOut() {

    sessionStorage.removeItem('username')

  }

}

**AuthGuard service:**

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

import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot } from '@angular/router';

import { AuthenticationService } from './authentication.service';

@Injectable({

  providedIn: 'root'

})

export class AuthGaurdService implements CanActivate{

  constructor(private router: Router,

    private authService: AuthenticationService) { }

  canActivate(route: ActivatedRouteSnapshot, state: RouterStateSnapshot) {

    if (this.authService.isUserLoggedIn())

      return true;

    this.router.navigate(['login']);

    return false;

  }

}

**Basic AuthInterceptor:**

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

import { HttpInterceptor, HttpRequest, HttpHandler } from '@angular/common/http';

@Injectable({

  providedIn: 'root'

})

export class BasicAuthHtppInterceptorService implements HttpInterceptor {

  constructor() { }

  intercept(req: HttpRequest<any>, next: HttpHandler) {

    if (sessionStorage.getItem('username') && sessionStorage.getItem('basicauth')) {

      req = req.clone({

        setHeaders: {

          Authorization: sessionStorage.getItem('basicauth')

        }

      })

    }

    return next.handle(req);

  }

}

**Login:**

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

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

import { AuthenticationService } from '../authentication.service';

@Component({

  selector: 'app-login',

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

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

})

export class LoginComponent implements OnInit {

  username = ''

  password = ''

  invalidLogin = false

  constructor(private router: Router,

    private loginservice: AuthenticationService) { }

  ngOnInit() {

  }

  checkLogin() {

    (this.loginservice.authenticate(this.username, this.password).subscribe(

      data => {

        this.router.navigate([''])

        this.invalidLogin = false

      },

      error => {

        this.invalidLogin = true

      }

    )

    );

  }

}

**Logout:**

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

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

import { AuthenticationService } from '../authentication.service';

@Component({

  selector: 'app-logout',

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

  styleUrls: ['./logout.component.css']

})

export class LogoutComponent implements OnInit {

  constructor(

    private authentocationService: AuthenticationService,

    private router: Router) {

  }

  ngOnInit() {

    this.authentocationService.logOut();

    this.router.navigate(['login']);

  }

}