**Online Bank Management Application**

**Tools and Technologies**

* Front-End: Html, CSS, bootstrap, Angular 8
* Frameworks: Spring boot, Spring MVC, Spring Data, Spring security.
* Technology/Domain: Java
* Database: MYSQL
* Server: Tomcat 9.5

**Objective of Bank Management application**

The bank is a major industry and still at some stage banks are doing some manual processes. So it’s really good to automate some manual processes that can help bankers and customers.

So this banking management system is a web application to manage bank activities. Where we can manage the customers,bank employees and Transection histories, Also a customer can do online payments.

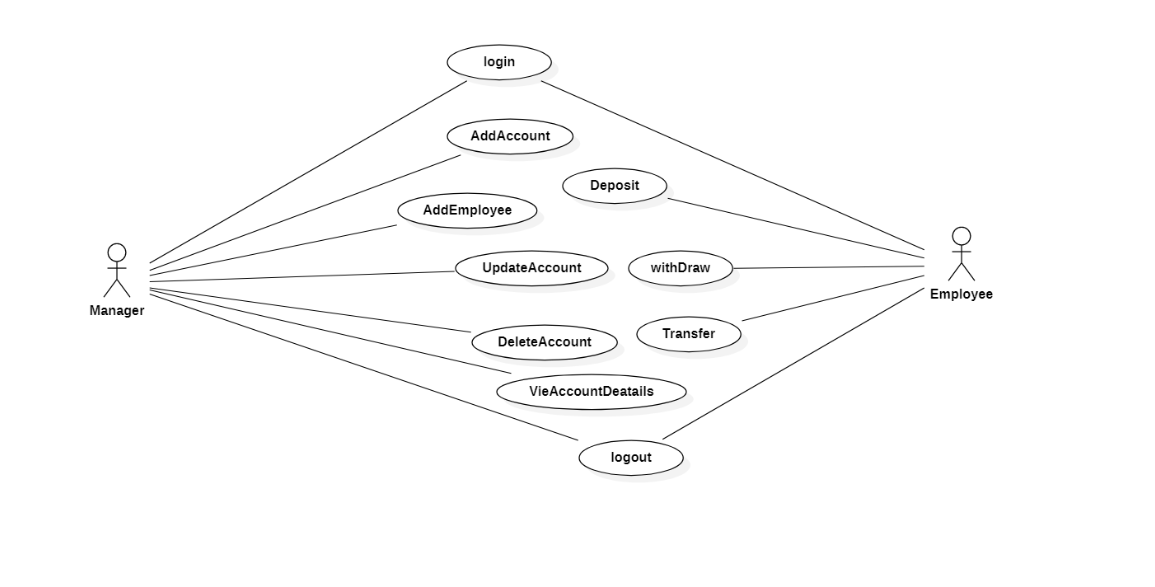
**There are two types of actors in the application. Bank Manager, Bank Employee**

**Bank Manager:**

Manager is the root user of the application. The is the admin of the application

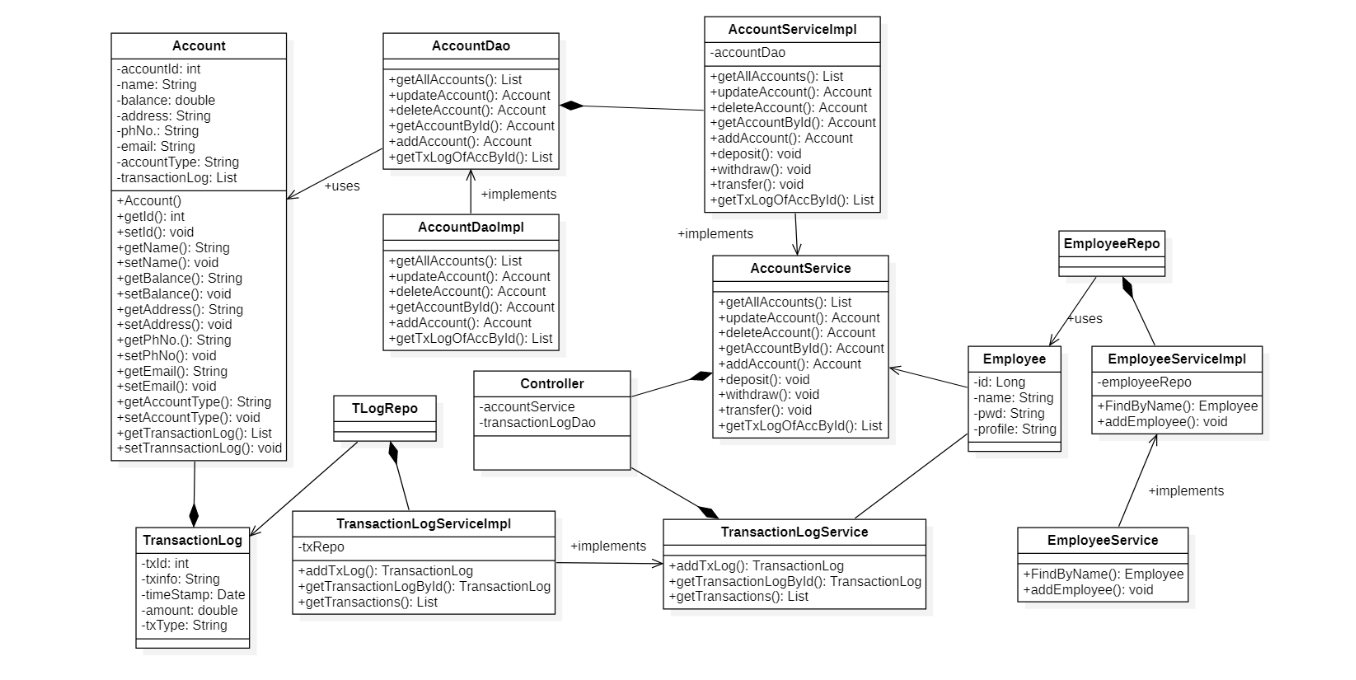
**He/she can do following activities:**

* The manager can ADD/EDIT/UPDATE bank accounts to the bank.
* manager can ADD/EDIT/UPDATE bank employee
* Bank manager authorized an bank employee (clerk) to do bank transactions
* Bank manager will manage all the online payment logs.
* When a new user will be added to the bank. A unique account number will generate automatically for the customer that the user will use during the transactions.



**Bank Employee:**

* The bank employee can do transactions as per the customer’s request.
* He/she can transfer fund, deposit amount, withdraw amount for an customer
* He can print customer transaction details on customer passbook



**Angular Code and Output Screens:**

**Account Component:**

export class AccountComponent implements OnInit {

  accounts: Bankaccount[];

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

  ngOnInit() {

    this.getAccounts();

  }

  private getAccounts(){

    this.accountService.getAccountsList().subscribe(data=>{

  this.accounts=data;

    });

  }

  updateAccount(id: number){

    console.log(`-----------`)

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

  }

  deleteAccount(id: number){

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

      this.getAccounts();

      console.log(data);

    })

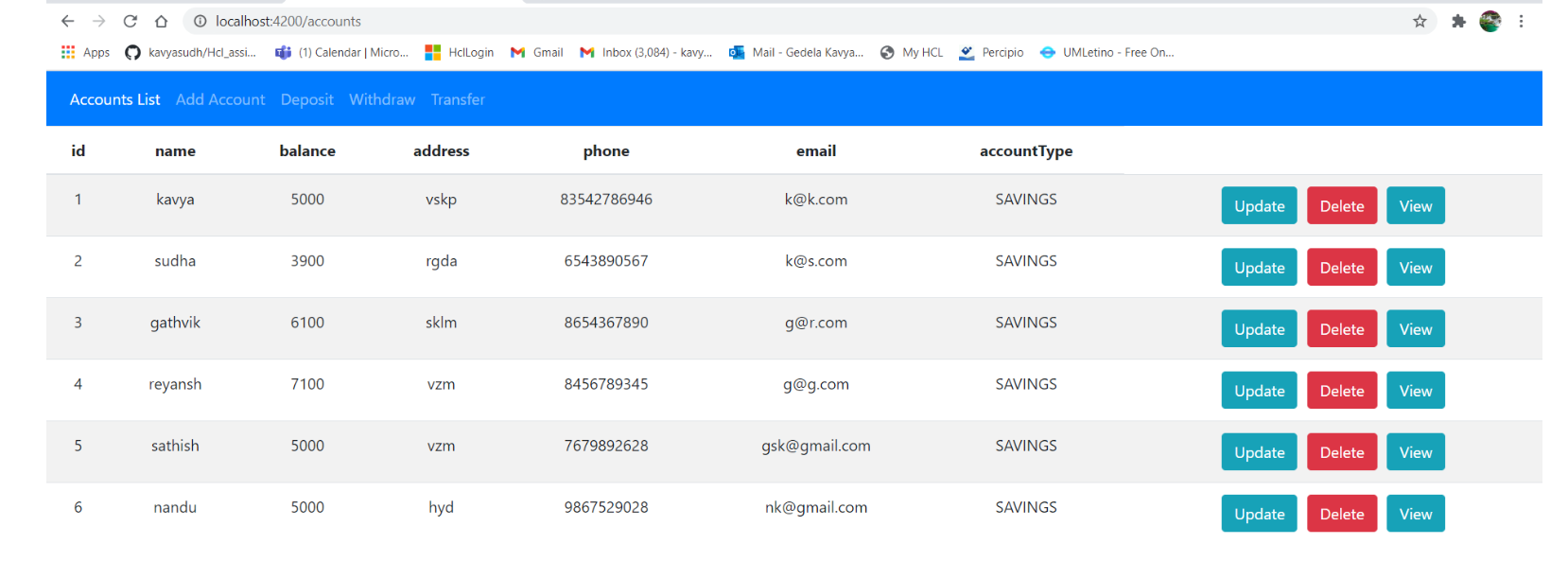
  }

  accountDetails(id: number){

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

  }

}



**Create-Account Compoment:**

export class CreateAccountComponent implements OnInit {

  bankaccount:Bankaccount = new Bankaccount();

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

  ngOnInit() : void {

  }

  saveAccount(){

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

      console.log(data);

      this.goToAccountList();

    },

    error => console.log(error));

  }

  goToAccountList(){

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

  }

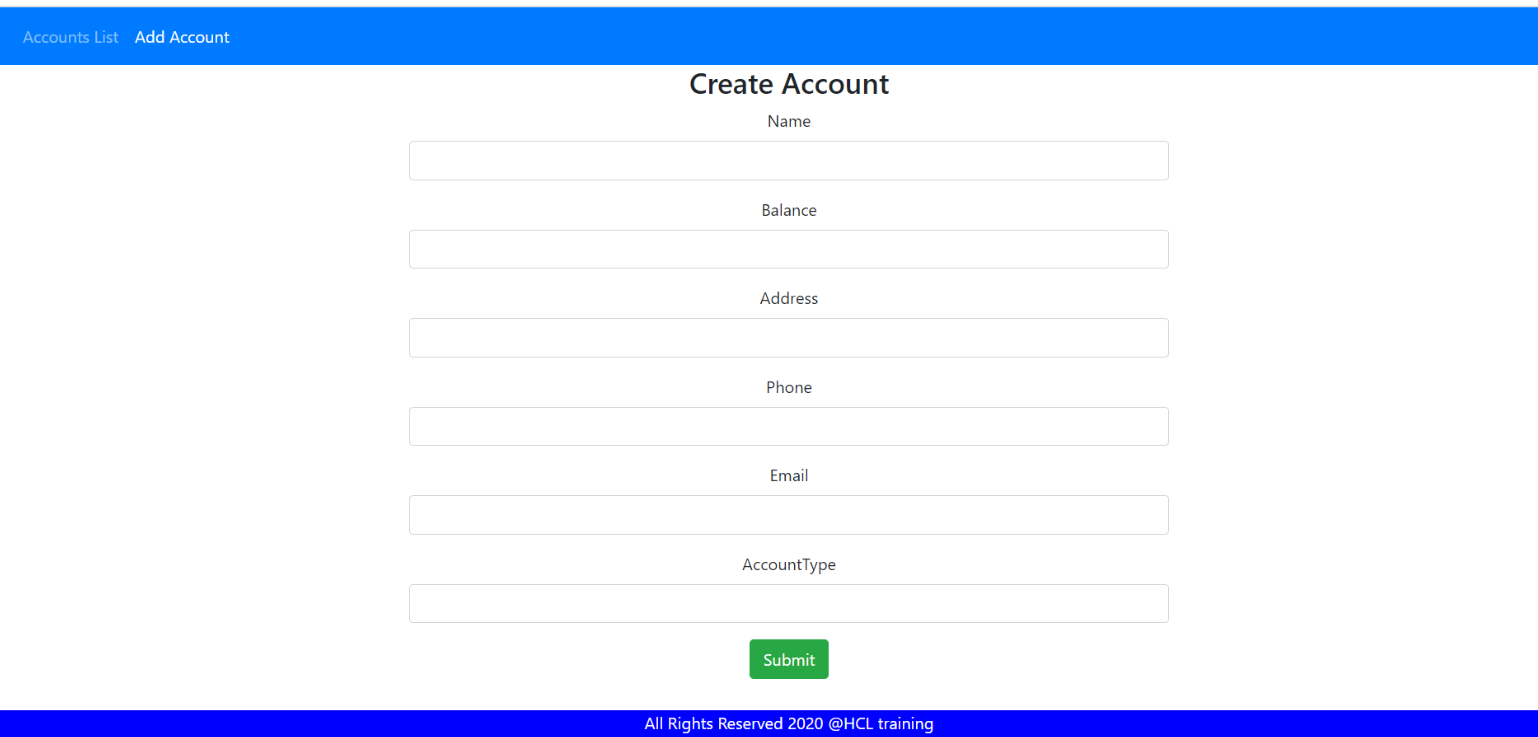
  onSubmit(){

    console.log(this.bankaccount);

    this.saveAccount();

  }

}



**Update-Account Component:**

export class UpdateAccountComponent implements OnInit {

  id: number;

  account: Bankaccount = new Bankaccount();

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

  ngOnInit(): void {

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

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

    this.account=data;

    }, error=>console.log(error))

  }

  onSubmit(){

    this.accountService.updateAccount(this.id, this.account)

    .subscribe(data=> {

        this.goToAccountList();

    }, error=> console.log(error))

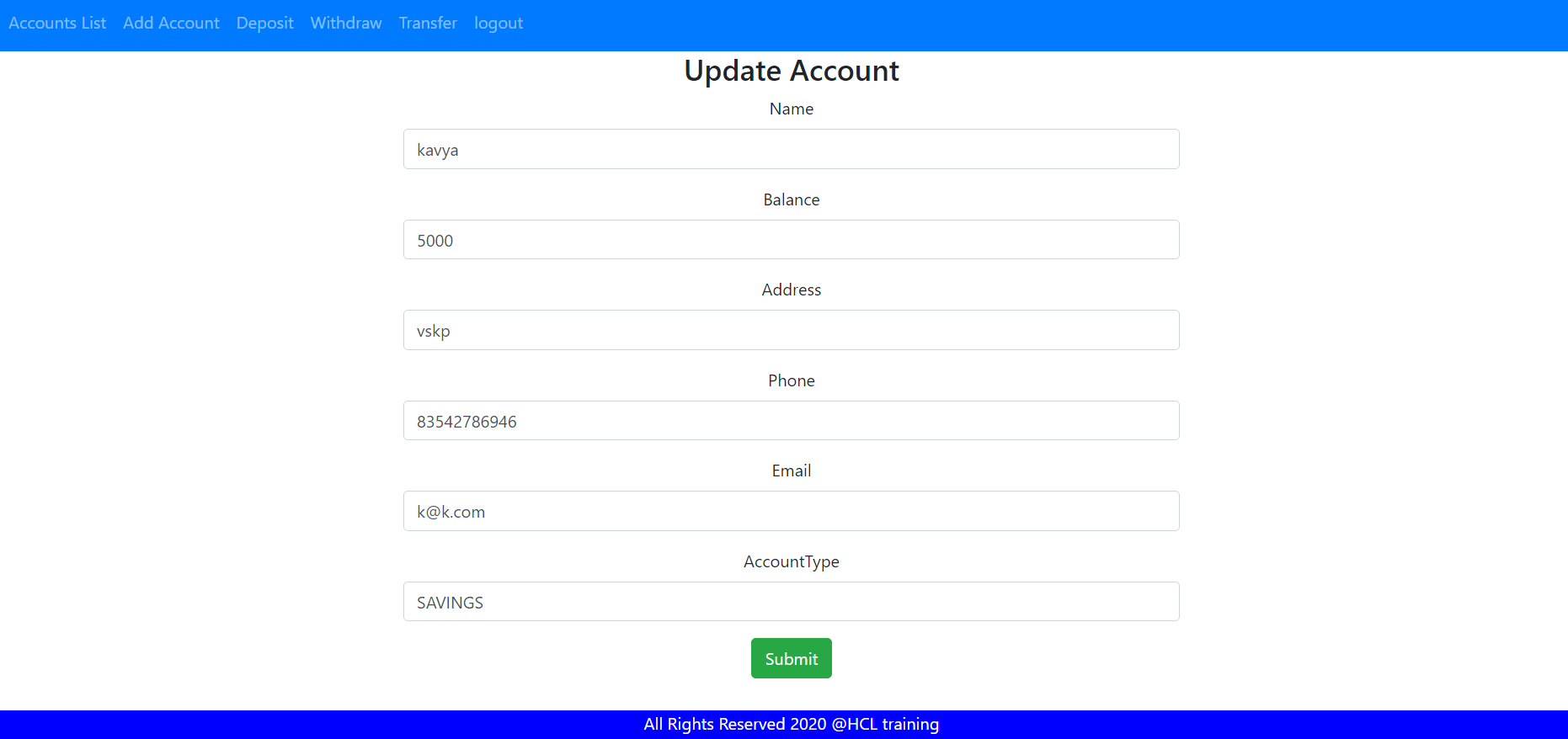
  }

  goToAccountList(){

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

  }

}



**Account Service:**

export class AccountService {

  private baseURL="http://localhost:8081/bankapp/account";

  constructor(private httpClient: HttpClient) { }

  getAccountsList(): Observable<Bankaccount[]>{

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

  }

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

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

  }

  updateAccount(id: number, account: Bankaccount): Observable<Object>{

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

  }

  getAccountById(id: number): Observable<Bankaccount>{

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

  }

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

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

  }

  deposit(deposit: Deposit): Observable<any>{

    return this.httpClient.post(`http://localhost:8081/account/deposit`, deposit);

  }

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

    return this.httpClient.post(`http://localhost:8081/account/withdraw`, withdraw);

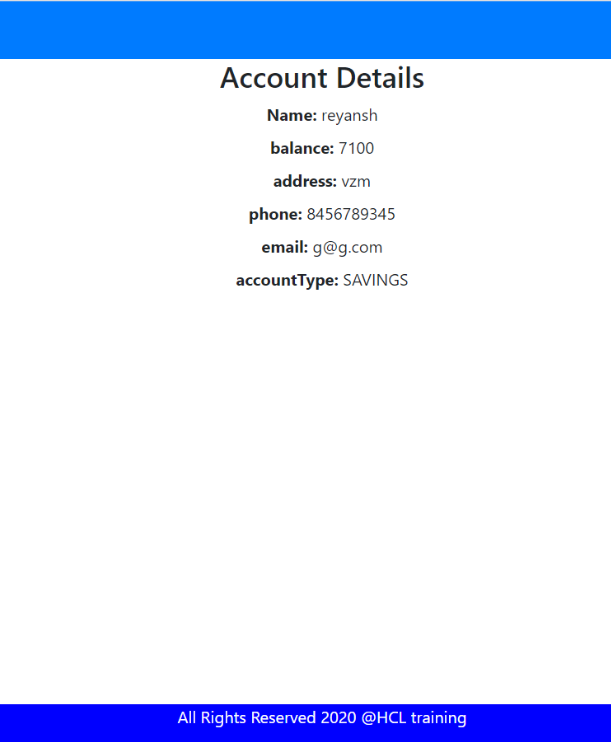
  }

  transfer(transfer: Transfer): Observable<any>{

    return this.httpClient.post(`http://localhost:8081/account/transfer`, transfer);

  }

}



export class Deposit {

    accountId: number;

    amount: string;

}

export class Withdraw {

    accountId: number;

    amount: string;

}

export class Transfer {

    fromAccountId:number;

    toAccountId: number;

    amount: string;

}

deposit(deposit: Deposit): Observable<any>{

    return this.httpClient.post(`http://localhost:8081/account/deposit`, deposit);

  }

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

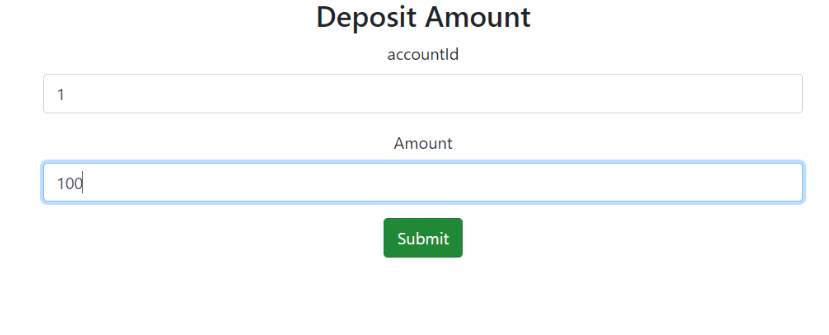
    return this.httpClient.post(`http://localhost:8081/account/withdraw`, withdraw);

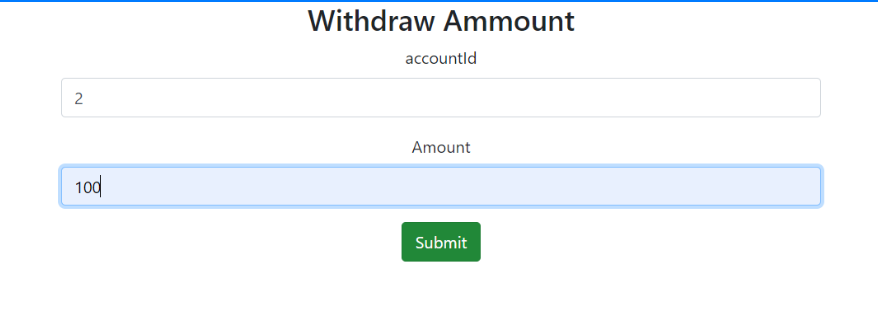
  }

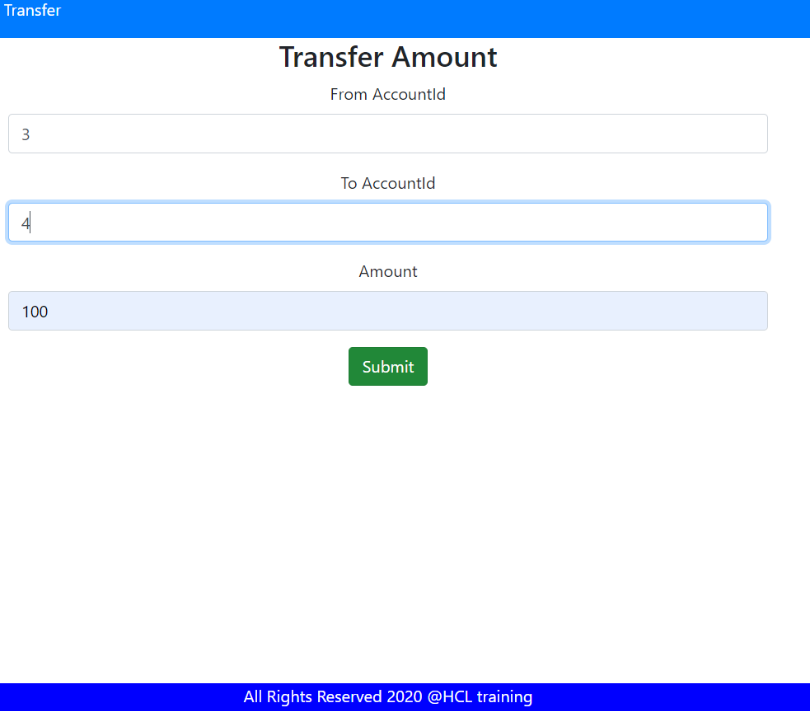
  transfer(transfer: Transfer): Observable<any>{

    return this.httpClient.post(`http://localhost:8081/account/transfer`, transfer);

  }







**Login Component:**

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

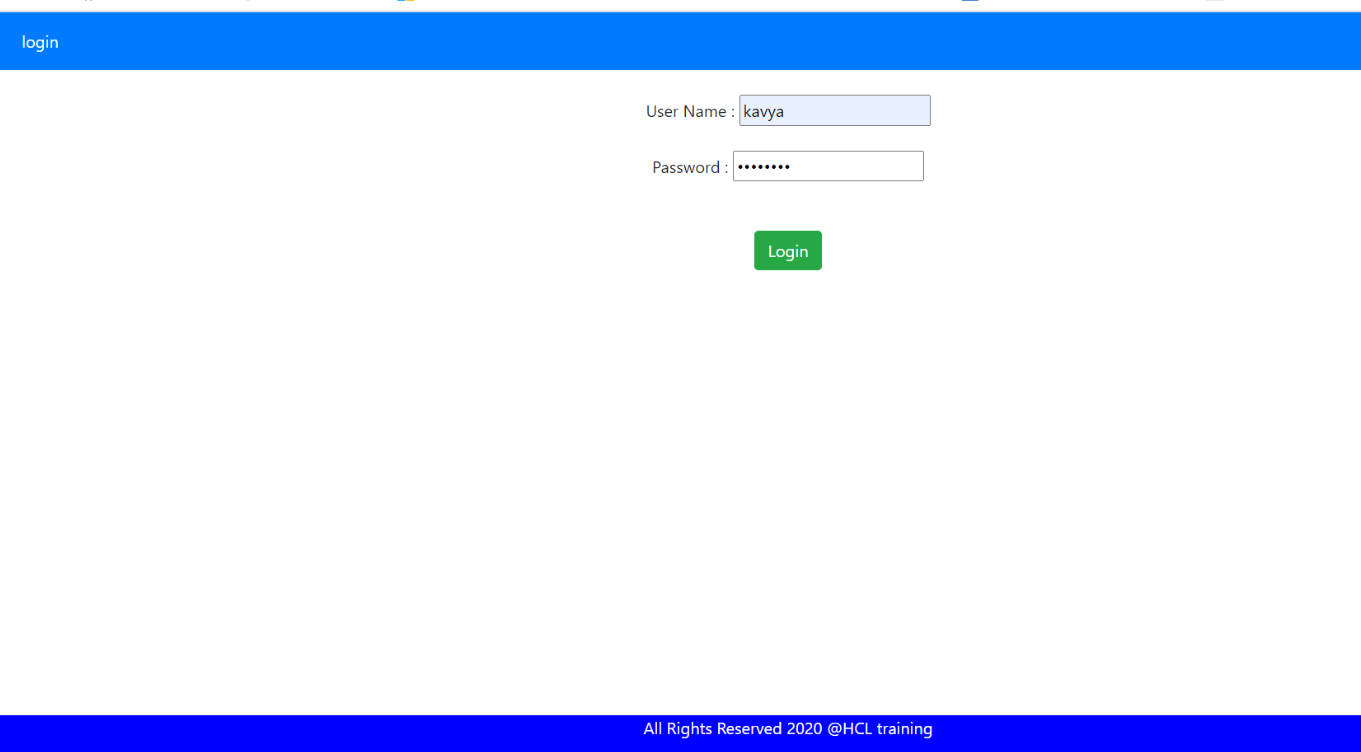
      }

    )

    );

  }

}



**Logout Component:**

export class LogoutComponent implements OnInit {

  constructor(

    private authentocationService: AuthenticationService,

    private router: Router) {

  }

  ngOnInit() {

    this.authentocationService.logOut();

    this.router.navigate(['login']);

  }

}

