**BANK APPLICATION**

**Types:** There are three types of users for this application..

1. Manager.
2. Employee.
3. Customer

**Users and their roles:**

1. **Manager** :

Manager is the root user of the application. His/her is the admin of the application, Admin can perform all actions on an account. The actions include creating new account, create new user for the application, suspend a account and delete a account.

1. **Employee**:

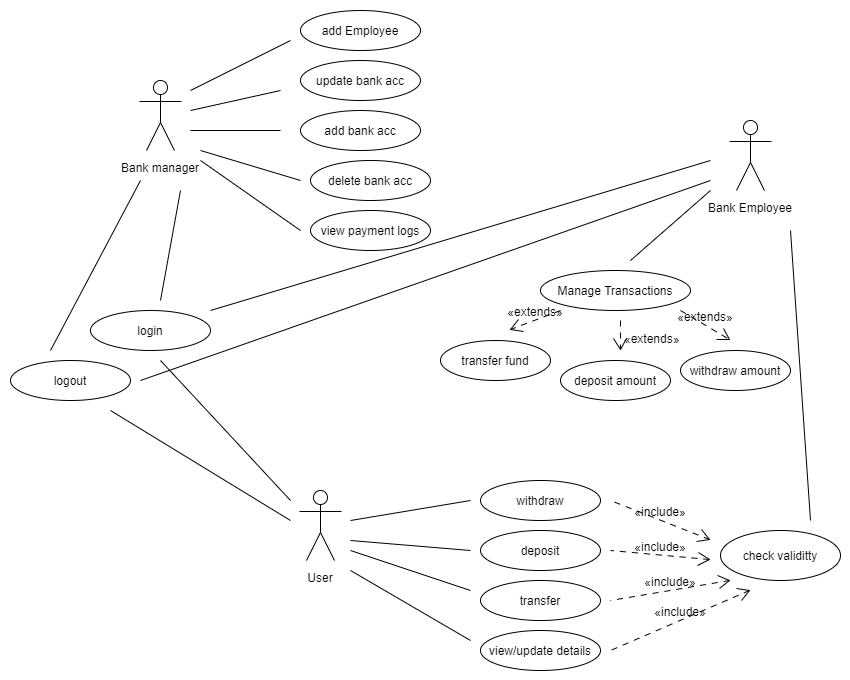
Employee is also a user who has limited actions. The bank employee can do transactions as per the customer’s request. He/she can transfer fund, deposit amount, withdraw amount for an customer.

1. **Customer**:

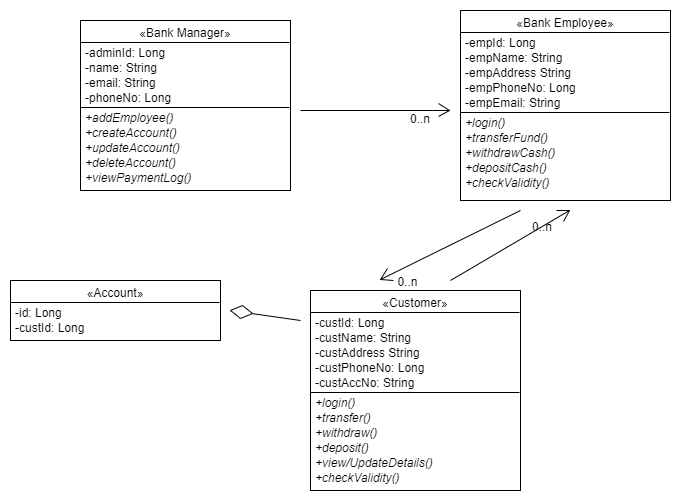
Customer can login into the application once approved by bank manager. He/she can check his bank balance and transaction history. Customer can send money to another Customer. The Customer can update his address details subjected to manager approval.

**Diagrams**:

Use Case diagram:



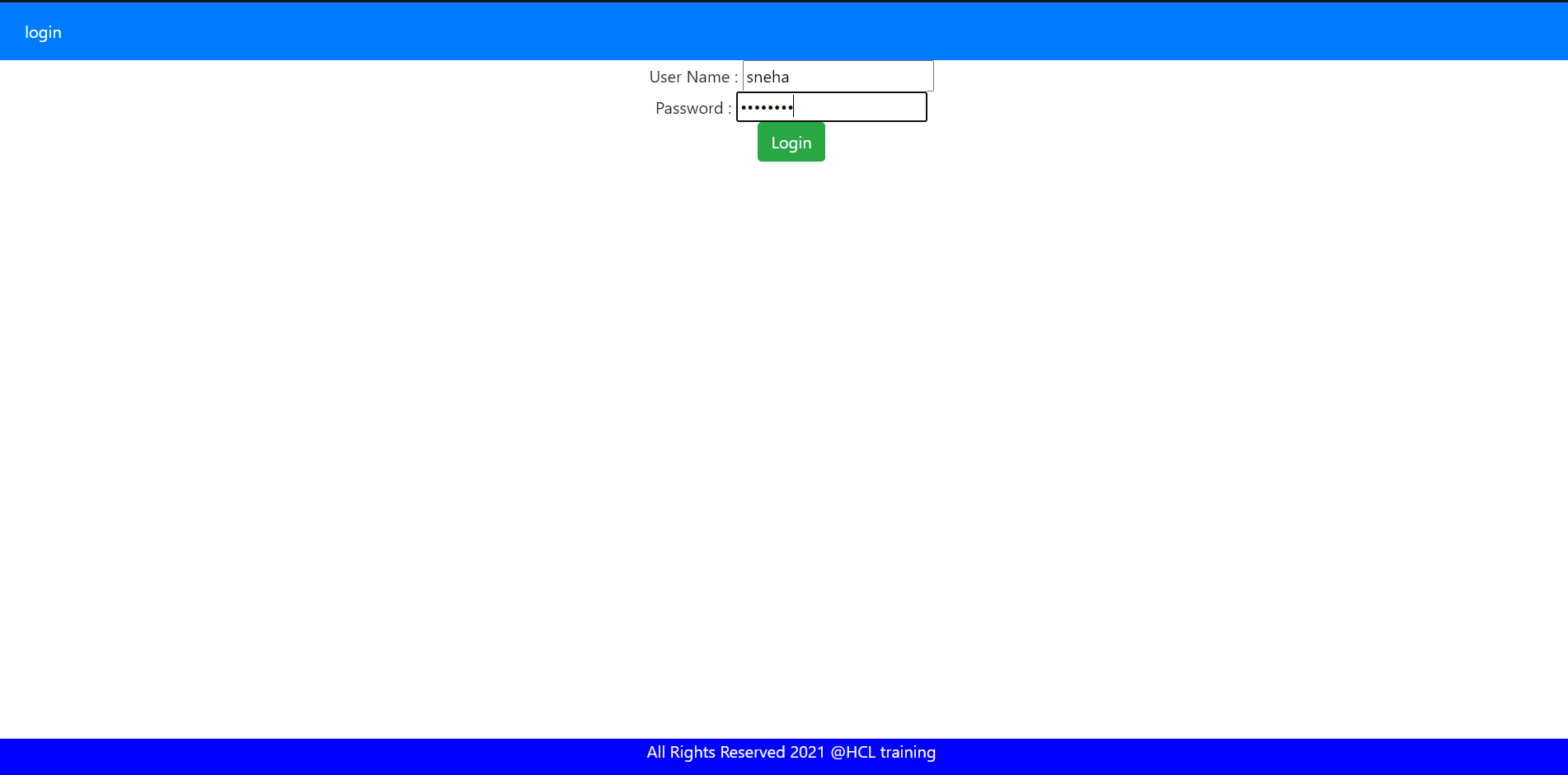
Class Diagram:



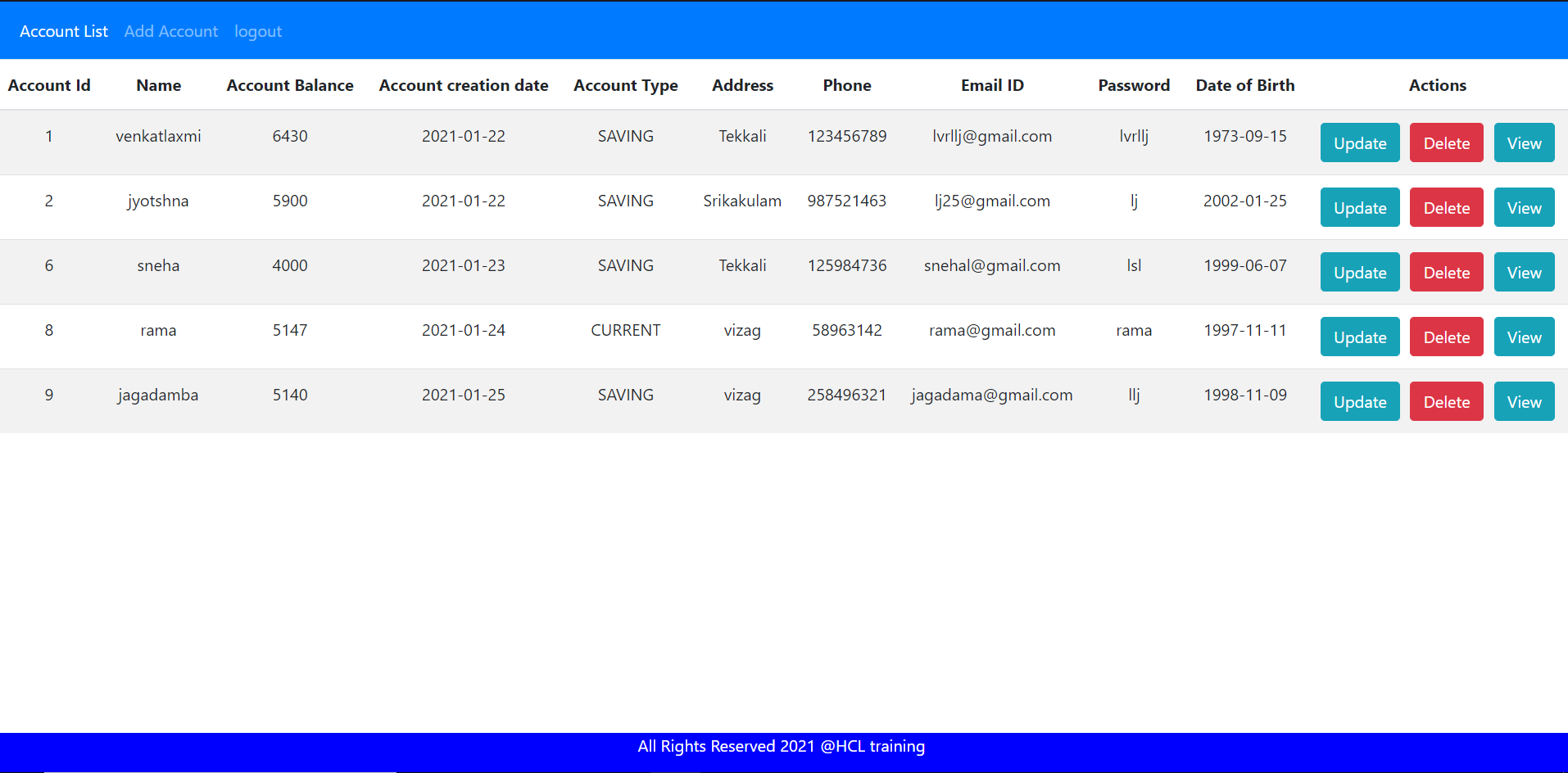
**Actions:**

**1.Login**: This is used to login for particular person. It takes parameters like username and password. If login fails he/she will be redirected to home page.

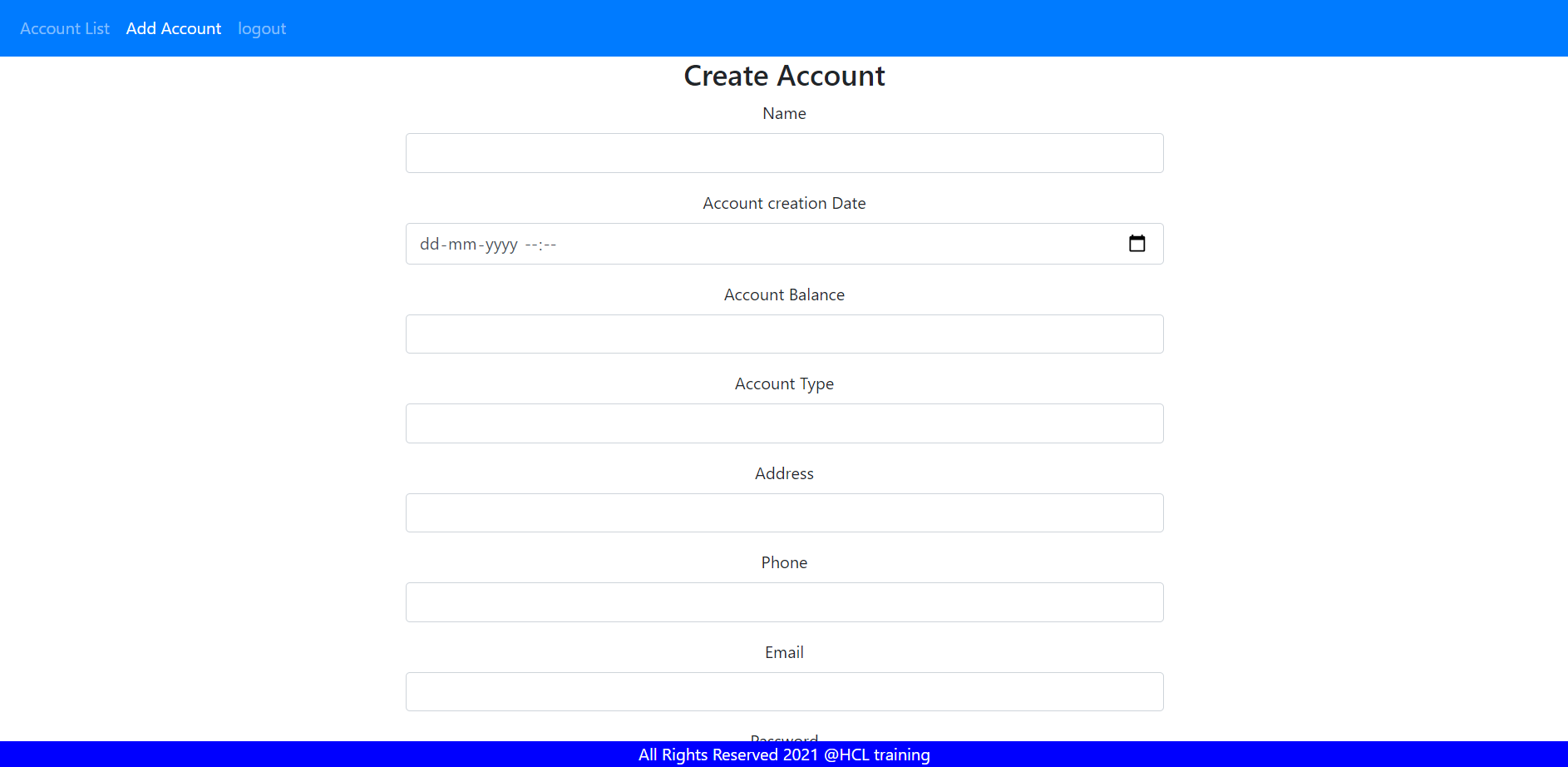
**Output:**

****

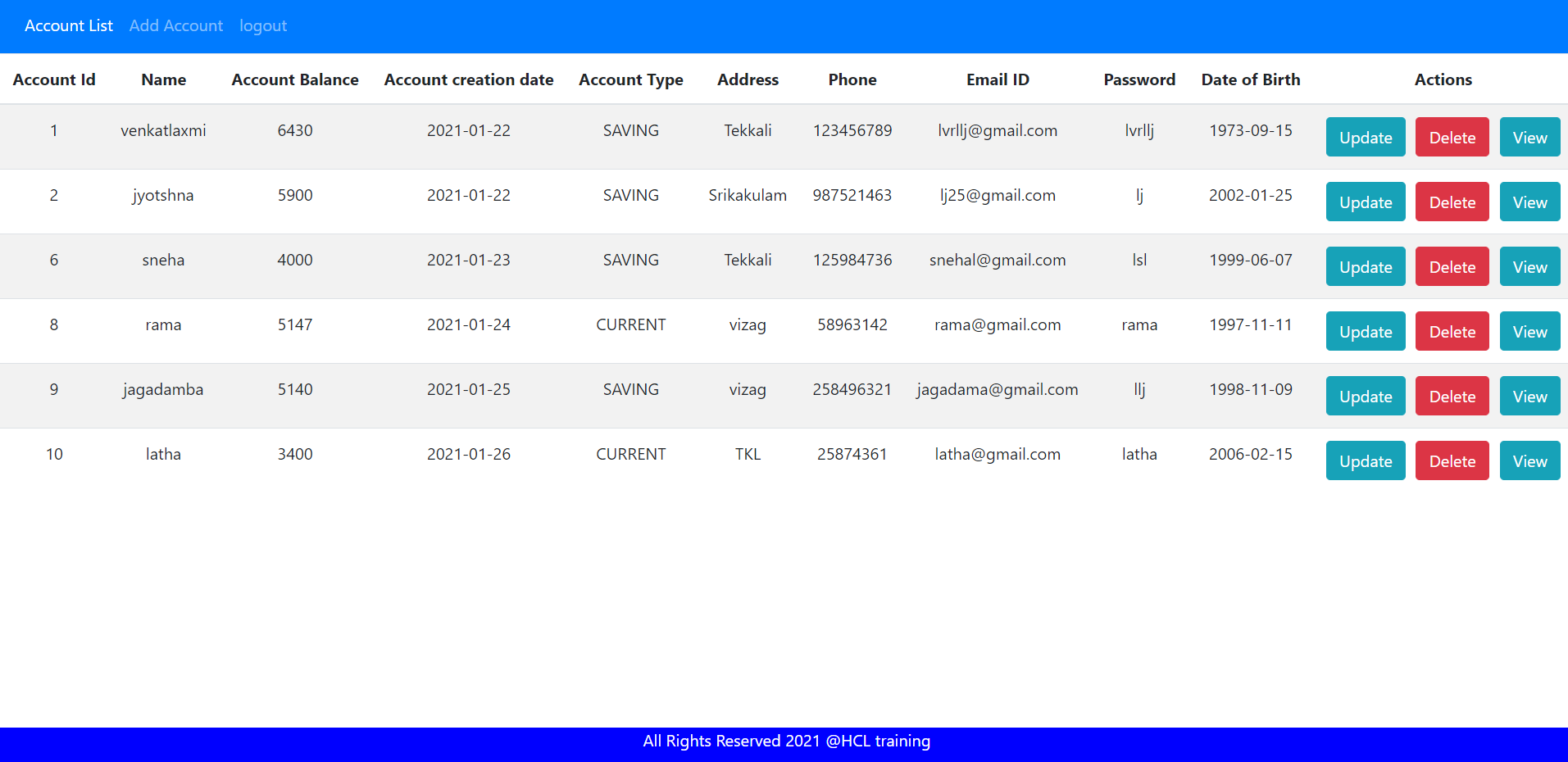
2. **Account List:** After the user Log In to the application with his/her credential the total accounts list are given below.



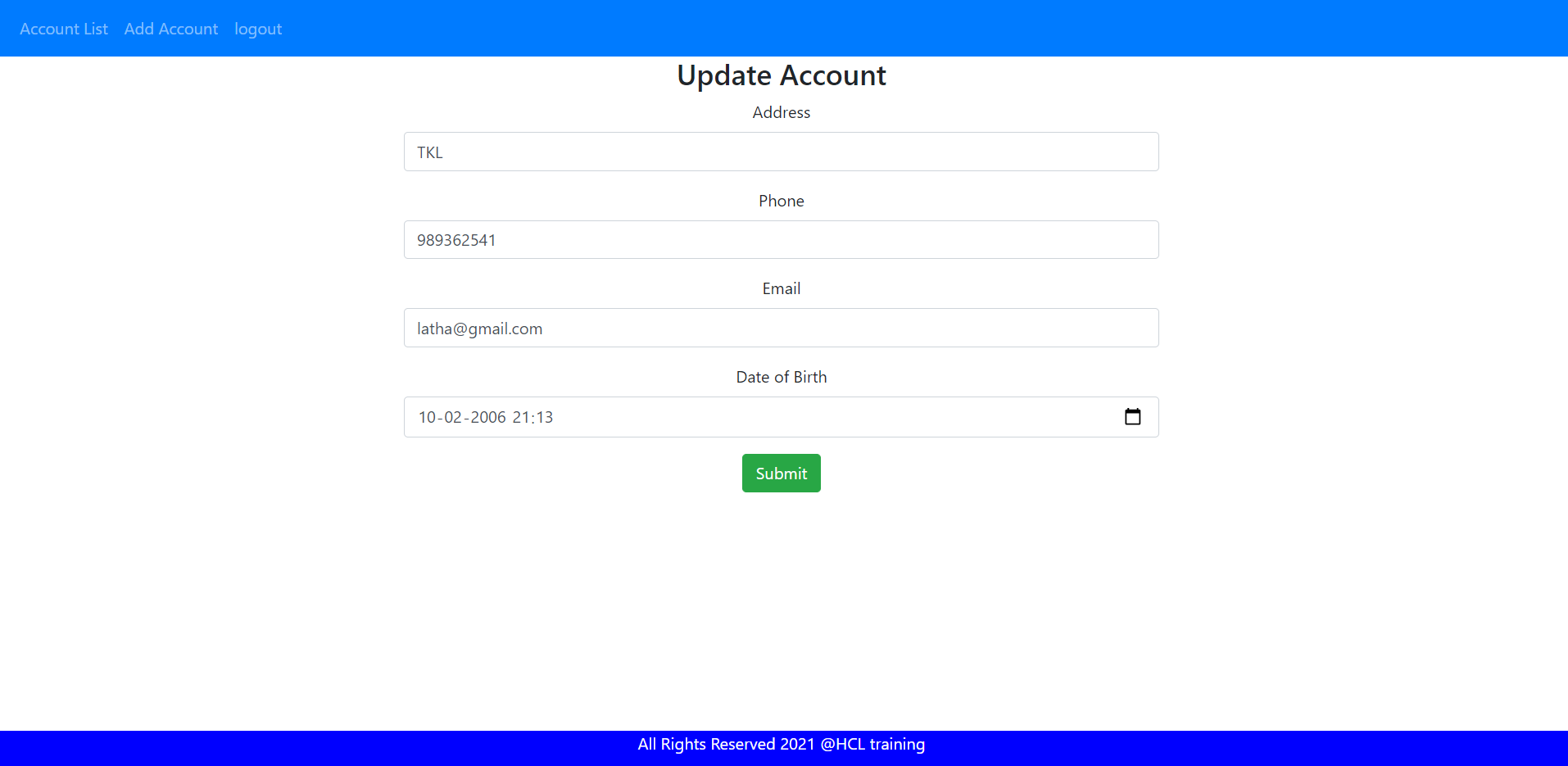
**3. Create Account:** This action allows us to add new Customer. The inputs are username, Email, address, phone number, Date of Birth.



4. After adding new account to customer to view whether they are added or not click on account list to view total accounts list

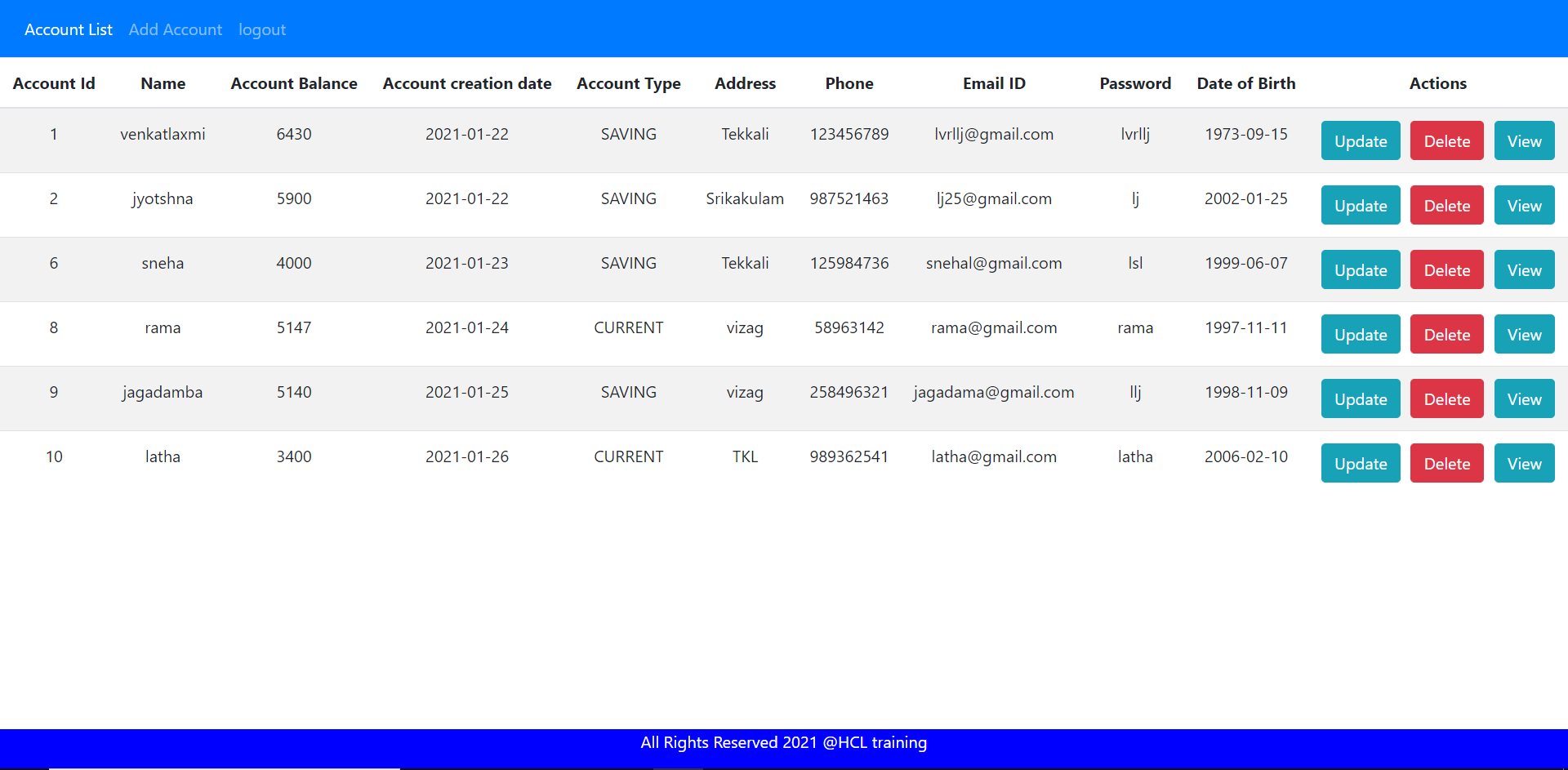


**5. Update account:** This action is used to update account details of particular customer we can update details like address, mobile number, email id.

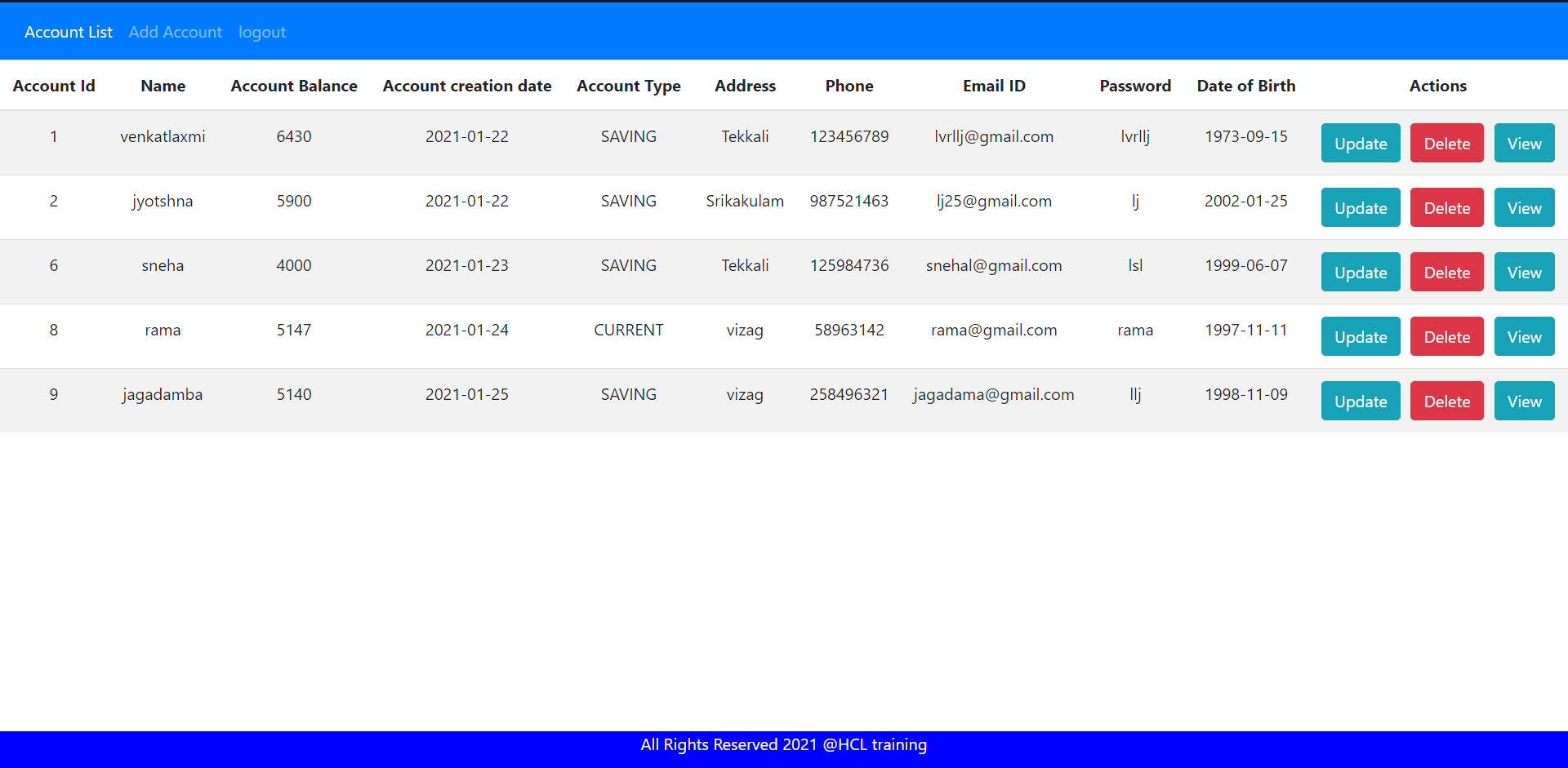


**6.view account details** :after updating particular account: click on account list then we

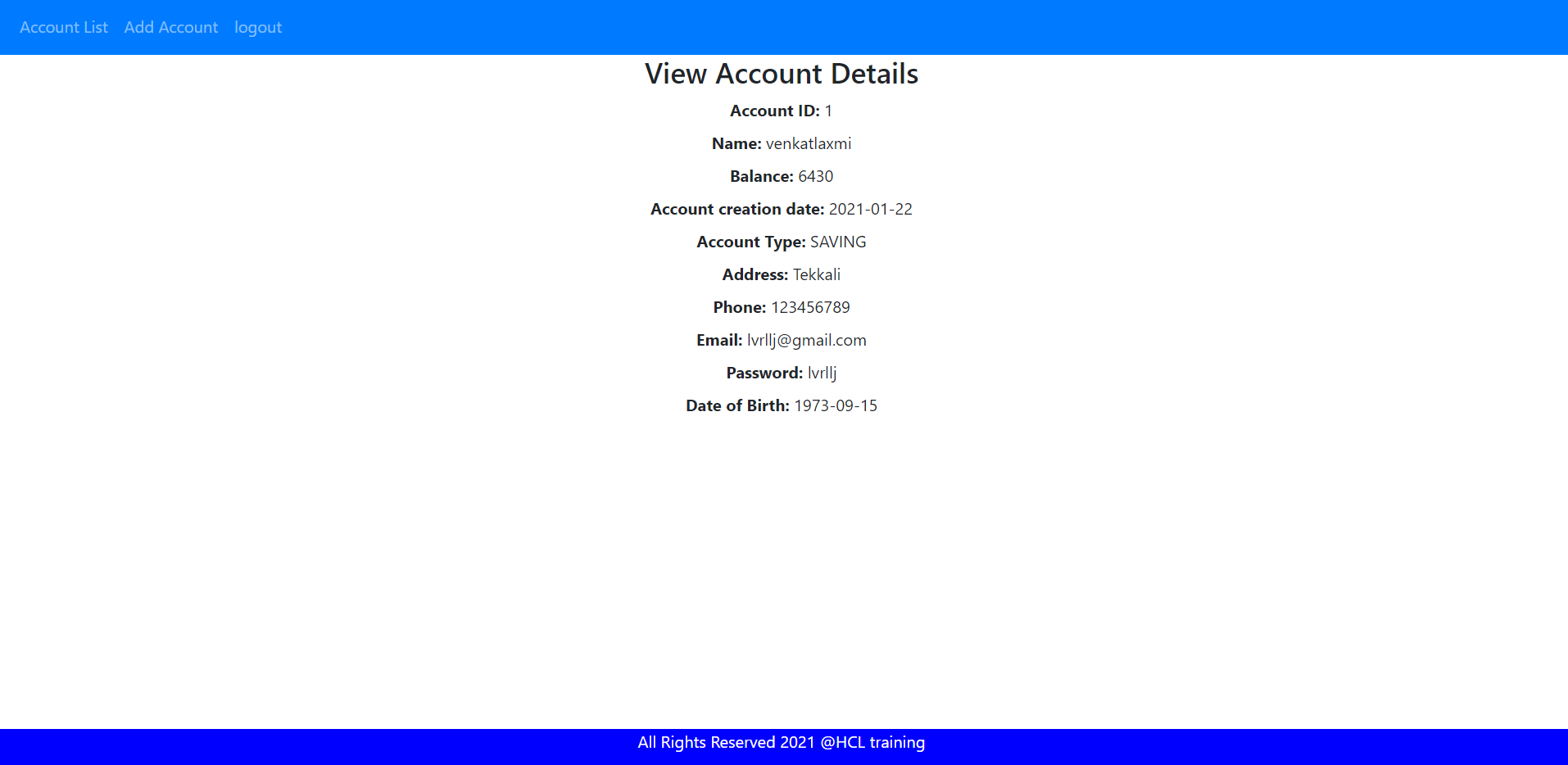
can view all account details with the help of account id we can know particular account details whether they are updated or not.



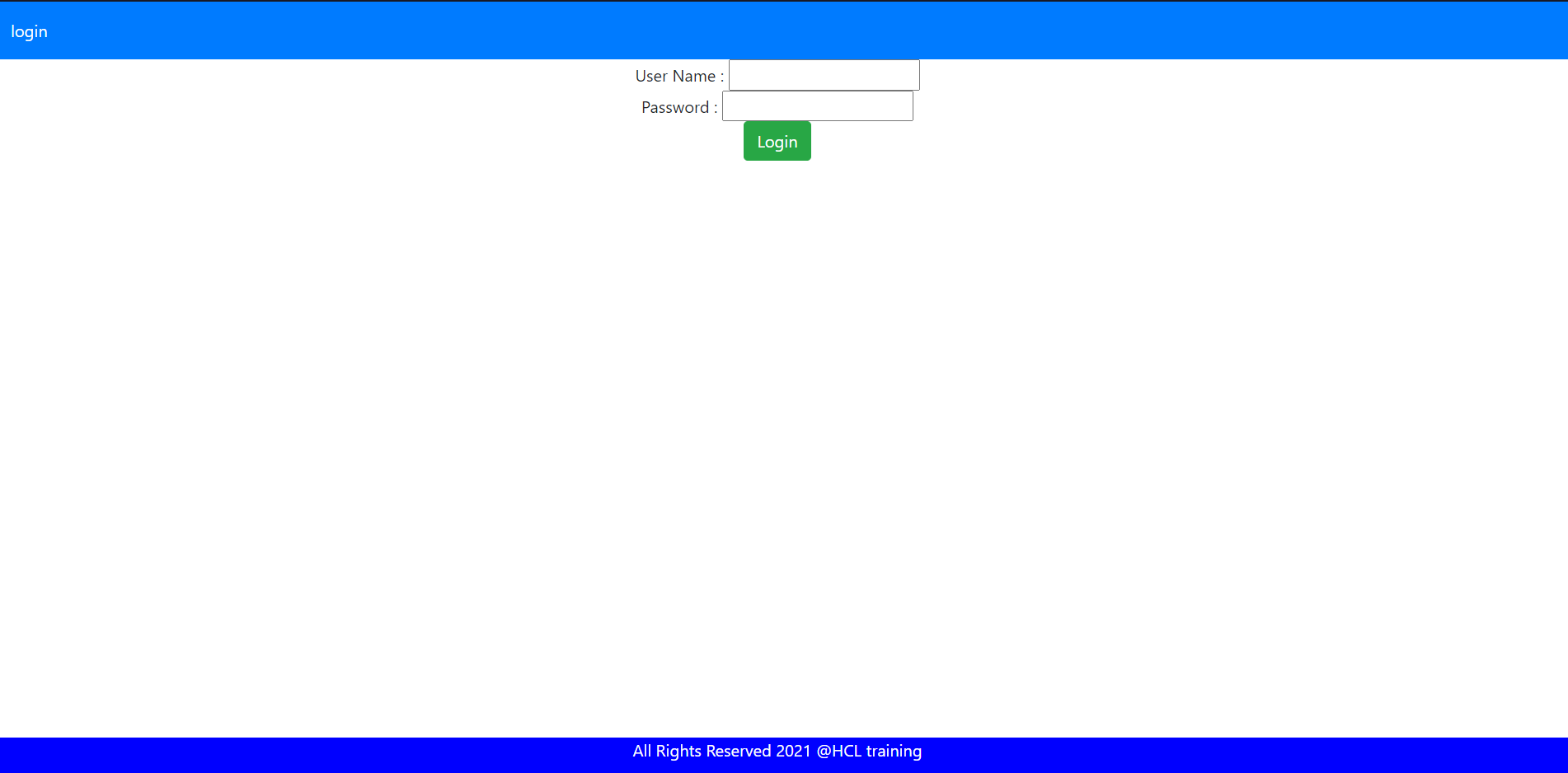
**7.Delete Account:** This action is used to delete particular customer account.By the use of delete button as shown below we can delete account.



**8. Viewing account details :** To view particular account details of the customer we can view it by clicking view button of the particular customer

****

**9. logout:** This action is used to logout the application. After clicking logout option then it will redirect to login page.



**Code:**

**Account:**

@Entity

@Table(name = "account\_details")

**public** **class** Account {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

**private** **int** accId;

**private** **double** balance;

**private** LocalDate accountCreationDate;

**private** String accountType;

**private** String name;

**private** String address;

**private** String phone;

**private** String email;

**private** String password;

**private** LocalDate dob;

@JsonIgnore

@JoinColumn(name = "accId\_fk", nullable = **false**)

@OneToMany(cascade = CascadeType.***ALL***)

**private** List<TransactionLog> transLog = **new** ArrayList<>();

**Account Service:**

**public** **interface** AccountService {

**public** List<Account> getAllAcounts();

**public** Account getAccountById(**int** accountId);

**public** Account addAccount(Account account);

**public** Account updateAccount(**int** accountId, Account account);

**public** Account deleteAccount(**int** accountId);

**public** **void** transfer(**int** fromId, **int** toId, **double** amount);

**public** **void** deposit(**int** accountId, **double** amount);

**public** **void** withdraw(**int** accountId, **double** amount);

**public** List<TransactionLog> getAllTransactions();

}

**Account Service Implementation:**

package com.bankapp.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.bankapp.dao.AccountRepo;

import com.bankapp.dao.TransactionLogRepo;

import com.bankapp.entities.Account;

import com.bankapp.entities.TransactionLog;

import com.bankapp.exceptions.ResourceNotFoundException;

@Service

@Transactional

public class AccountServiceImpl implements AccountService {

private AccountRepo accountRepo;

private TransactionLogRepo transLogRepo;

@Autowired

public AccountServiceImpl(AccountRepo accountRepo, TransactionLogRepo transLogRepo) {

this.accountRepo = accountRepo;

this.transLogRepo = transLogRepo;

}

@Override

public List<Account> getAllAcounts() {

return (List<Account>) accountRepo.findAll();

}

@Override

public Account getAccountById(int accountId) {

return accountRepo.findById(accountId)

.orElseThrow(() -> new ResourceNotFoundException("account with id " + accountId + "not found"));

}

@Override

public Account addAccount(Account account) {

return accountRepo.save(account);

}

@Override

public Account updateAccount(int accountId, Account account) {

Account accToUpdate = getAccountById(accountId);

accToUpdate.setAddress(account.getAddress());

accToUpdate.setDob(account.getDob());

accToUpdate.setEmail(account.getEmail());

accToUpdate.setPhone(account.getPhone());

return accountRepo.save(accToUpdate);

}

@Override

public Account deleteAccount(int accountId) {

Account accToDelete = getAccountById(accountId);

accountRepo.delete(accToDelete);

return accToDelete;

}

@Override

public void transfer(int fromId, int toId, double amount) {

Account fromAcc = getAccountById(fromId);

Account toAcc = getAccountById(toId);

fromAcc.setBalance(fromAcc.getBalance() - amount);

toAcc.setBalance(toAcc.getBalance() + amount);

TransactionLog fromAccLog = new TransactionLog("withdraw", fromId, toId, "withdraw");

fromAcc.getTransLog().add(fromAccLog);

TransactionLog toAccLog = new TransactionLog("deposit", fromId, toId, "deposit");

toAcc.getTransLog().add(toAccLog);

accountRepo.save(fromAcc);

accountRepo.save(toAcc);

}

@Override

public void deposit(int accountId, double amount) {

Account toDeposit = getAccountById(accountId);

toDeposit.setBalance(toDeposit.getBalance() + amount);

TransactionLog accLog = new TransactionLog("deposit", 0, accountId, "deposit");

toDeposit.getTransLog().add(accLog);

accountRepo.save(toDeposit);

}

@Override

public void withdraw(int accountId, double amount) {

Account toWithdraw = getAccountById(accountId);

toWithdraw.setBalance(toWithdraw.getBalance() - amount);

TransactionLog accLog = new TransactionLog("withdraw", accountId, 0, "withdraw");

toWithdraw.getTransLog().add(accLog);

accountRepo.save(toWithdraw);

}

@Override

public List<TransactionLog> getAllTransactions() {

return transLogRepo.findAll();

}

}

**Controller:**

package com.bankapp.controller;

import java.util.List;

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

import org.springframework.http.MediaType;

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.dao.AuthResponse;

import com.bankapp.dto.DepositAmount;

import com.bankapp.dto.TransferAmount;

import com.bankapp.dto.WithdrawAmount;

import com.bankapp.entities.Account;

import com.bankapp.entities.TransactionLog;

import com.bankapp.service.AccountService;

@CrossOrigin(origins = "http://localhost:4200")

@RestController

public class AccountRestController {

@Autowired

private AccountService accountService;

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

public List<Account> allAccounts() {

return accountService.getAllAcounts();

}

@GetMapping(produces = "application/json")

@RequestMapping({ "/validateLogin" })

public AuthResponse validateLogin() {

return new AuthResponse("User successfully authenticated");

}

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

public List<TransactionLog> allTransactions() {

return accountService.getAllTransactions();

}

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

public Account getAccountById(@PathVariable(name = "accId") int accId) {

return accountService.getAccountById(accId);

}

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

public Account addNewAccount(@RequestBody Account account) {

return accountService.addAccount(account);

}

@PutMapping(path = "account/{accId}", produces = MediaType.APPLICATION\_JSON\_VALUE, consumes = MediaType.APPLICATION\_JSON\_VALUE)

public Account updateAccount(@PathVariable(name = "accId") int accId, @RequestBody Account account) {

return accountService.updateAccount(accId, account);

}

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

public Account deleteAccount(@PathVariable(name = "accId") int accId) {

return accountService.deleteAccount(accId);

}

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

public String transferFund(@RequestBody TransferAmount transAmt) {

accountService.transfer(transAmt.getFromId(), transAmt.getToId(), transAmt.getAmount());

return "fund is transferred";

}

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

public String depositFund(@RequestBody DepositAmount depositAmt) {

accountService.deposit(depositAmt.getAccountId(), depositAmt.getAmount());

return "amount deposited successfully";

}

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

public String withdrawFund(@RequestBody WithdrawAmount withdrawAmt) {

accountService.withdraw(withdrawAmt.getAccountId(), withdrawAmt.getAmount());

return "amount withdrawl successfully";

}

}

**Security Configuration:**

**package** com.bankapp.config;

**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(AuthenticationManagerBuilder auth) **throws** Exception {

auth.inMemoryAuthentication().withUser("sanju").password("{noop}sanju123").roles("USER");

}

@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception {

http.csrf().disable().

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

.and().httpBasic();

}

}

**Angular Code:**

**Angular Code:**

**Account Class:**

export class Account {

    accId: number;

    name: string;

    balance: number;

    accountCreationDate: Date;

    accountType: string;

    address: string;

    phone: string;

    email: string;

    password: string;

    dob: Date;

}

**AccountComponent:**

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

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

import { Account } from '../account';

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

@Component({

  selector: 'app-account',

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

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

})

export class AccountComponent implements OnInit {

  accounts: Account[];

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

  ngOnInit() {

    this.getAccounts();

  }

  private getAccounts() {

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

      this.accounts = data;

    });

  }

  updateAccount(accId: number) {

    console.log(`-----------`)

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

  }

  deleteAccount(accId: number) {

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

      this.getAccounts();

      console.log(data);

    })

  }

  accountDetails(accId: number) {

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

  }

}

**Account service:**

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

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

import { Observable } from 'rxjs';

import { Account } from './account';

@Injectable({

  providedIn: 'root'

})

export class AccountService {

  private baseURL = "http://localhost:8080/bankapp/account";

  constructor(private httpClient: HttpClient) { }

  getAccountList(): Observable<Account[]> {

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

  }

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

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

  }

  getAccountById(accId: number): Observable<Account> {

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

  }

  updateAccount(accId: number, account: Account): Observable<Object> {

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

  }

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

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

  }

}

**Create-Account:**

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

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

import { Account } from '../account';

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: Account = new Account();

  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.name" name="name">

        </div>

        <div class="form-group">

            <label> Account creation Date </label>

            <input type="datetime-local" class="form-control" id="accountCreationDate"

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

        </div>

        <div class="form-group">

            <label> Account Balance </label>

            <input type="text" class="form-control" id="balance" [(ngModel)]="account.balance" name="balance">

        </div>

        <div class="form-group">

            <label> Account Type </label>

            <input type="text" class="form-control" id="accountType" [(ngModel)]="account.accountType"

                name="accountType">

        </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> Phone </label>

            <input type="text" class="form-control" id="phone" [(ngModel)]="account.phone" name="phone">

        </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> Password </label>

            <input type="text" class="form-control" id="password" [(ngModel)]="account.password" name="password">

        </div>

        <div class="form-group">

            <label> Date of Birth </label>

            <input type="datetime-local" class="form-control" id="dob" [(ngModel)]="account.dob" name="dob">

        </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 { Account } from '../account';

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

@Component({

  selector: 'app-update-account',

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

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

})

export class UpdateAccountComponent implements OnInit {

  accId: number;

  account: Account = new Account();

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

  ngOnInit(): void {

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

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

      this.account = data;

    }, error => console.log(error))

  }

  onSubmit() {

    this.accountService.updateAccount(this.accId, 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> Update Account </h3>

    <form (ngSubmit)="onSubmit()">

        <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> Phone </label>

            <input type="text" class="form-control" id="phone" [(ngModel)]="account.phone" name="phone">

        </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> Date of Birth </label>

            <input type="datetime-local" class="form-control" id="dob" [(ngModel)]="account.dob" name="dob">

        </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 { Account } from '../account';

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

@Component({

  selector: 'app-account-details',

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

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

})

export class AccountDetailsComponent implements OnInit {

  accId: number;

  account: Account = new Account();

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

  ngOnInit(): void {

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

    this.account = new Account();

    this.accountService.getAccountById(this.accId).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 \*ngIf="loginService.isUserLoggedIn()" routerLink="accounts" routerLinkActive="active"

        class="nav-link">Account List</a>

    </li>

    <li class="nav-item">

      <a \*ngIf="loginService.isUserLoggedIn()" routerLink="create-account" routerLinkActive="active"

        class="nav-link">Add Account</a>

    </li>

    <li class="nav-item">

      <a \*ngIf="!loginService.isUserLoggedIn()" routerLink="login" routerLinkActive="active" class="nav-link">login</a>

    </li>

    <li class="nav-item">

      <a \*ngIf="loginService.isUserLoggedIn()" 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 2021 @HCL training</span>

  </div>

</footer>

**App routing Module**

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

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

import { AccountDetailsComponent } from './account-details/account-details.component';

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

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

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

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

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

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

const routes: Routes = [

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

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

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

  { path: 'account-details/:accId', component: AccountDetailsComponent, 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:8080/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, UrlTree } from '@angular/router';

import { Observable } from 'rxjs';

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

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

@Injectable({

  providedIn: 'root'

})

export class BasicAuthHtppInterceptorService {

  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']);

  }

}