list.component.html

```
<h2>Admin {{a.username}}</h2>
<h3>Users list</h3>
<a routerLink="add" class="btn btn-sm btn-success mb-2">Add User</a>
<a routerLink="/pass" class="btn btn-link">change password</a>


<table class="table table-striped">
    <thead>
        <tr>
            <th style="width: 30%">Aadhar Id</th>
            <th style="width: 30%">Name</th>
            <th style="width: 30%">Phone</th>
            <th style="width: 10%"></th>
        </tr>
    </thead>
    <tbody>
        <tr *ngFor="let user of users">
            <td>{{user.id}}</td>
            <td>{{user.name}}</td>
            <td>{{user.price}}</td>
            <td style="white-space: nowrap">
                <a routerLink="issue/{{user.id}}" >issue Aadhar</a>
                <a routerLink="edit/{{user.id}}" class="btn btn-sm btn-primary me-1">Edit</a>
                <button (click)="deleteUser(user.id)" class="btn btn-sm btn-danger btn-delete-user"
[disabled]="user.isDeleting">
                    <span *ngIf="user.isDeleting" class="spinner-border spinner-border-sm"></span>
                    <span *ngIf="!user.isDeleting">Delete</span>
                </button>

            </td>
        </tr>
        <tr *ngIf="!users">
            <td colspan="4" class="text-center">
                <span class="spinner-border spinner-border-lg align-center"></span>
            </td>
        </tr>
    </tbody>
</table>
```

list.component.ts

```
import { Component, OnInit } from '@angular/core';
import { first } from 'rxjs/operators';

import { AccountService } from '@app/_services';

@Component({ templateUrl: 'list.component.html' })
export class ListComponent implements OnInit {

        users?: any[];
        a:any;

    constructor(private accountService: AccountService) {
        this.a=accountService.ad;
        console.log(this.a);
    }

    ngOnInit() {

        this.accountService.getAll()
```

```
            .pipe(first())
            .subscribe(users => this.users = users);
            console.log(this.users);
    }

    deleteUser(id: string) {
        const user = this.users!.find(x => x.id === id);
        user.isDeleting = true;
        this.accountService.delete(id)
            .pipe(first())
            .subscribe(() => this.users = this.users!.filter(x => x.id !== id));
    }
}
```

home.component.html

```
<div class="container">
        <h2>Welcome Admin</h2>
    <a class="btn btn-primary" id="view" routerLink="/users">View Users</a>
    <a routerLink="/pass" class="btn btn-link">change password</a>
</div>
```

home.component.ts

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

import { User } from '@app/_models';
import { AccountService } from '@app/_services';

@Component({ templateUrl: 'home.component.html' })
export class HomeComponent {
    user: User | null;

    constructor(private accountService: AccountService) {
        this.user = this.accountService.userValue;
    }
}
```

login.component.html

```
    <h4>Admin Login</h4>
        <form [formGroup]="form" (ngSubmit)="onSubmit()">

            <div >
                <label >username :</label>
                <input type="text" id="user" formControlName="username" [ngClass]="{ 'is-invalid':
submitted && forms.username.errors }" />
                <div *ngIf="submitted && forms.username.errors" class="invalid-feedback">
                    <div *ngIf="forms.username.errors.required">Username is required</div>
                </div>
            </div>

            <div >
                <label class="form-label">Password:</label>
                <input type="password" id="pass" formControlName="password" [ngClass]="{ 'is-invalid':
submitted && forms.password.errors }" />
                <div *ngIf="submitted && forms.password.errors" class="invalid-feedback">
                    <div *ngIf="forms.password.errors.required">Password is required</div>
                </div>
```

```
            </div>



            <div>
               <button  class="btn border-primary" id="login">Login</button>
            </div>
            <a routerLink="../register" class="btn btn-link"> Citizen Register</a>

            <a routerLink="../login" class="btn btn-link">Citizen Login</a>

        </form>
```

login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router, ActivatedRoute } from '@angular/router';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { first } from 'rxjs/operators';

import { AccountService, AlertService } from '@app/_services';

@Component({ templateUrl: 'login.component.html' })
export class LoginComponent implements OnInit {
    form!: FormGroup;
    submitted = false;

    constructor(
        private formBuilder: FormBuilder,
        private route: ActivatedRoute,
        private router: Router,
        private accountService: AccountService,
        private alertService: AlertService
    ) { }

    ngOnInit() {
        this.form = this.formBuilder.group({
            username: ['', Validators.required],
            password: ['', Validators.required]
        });
    }

    get forms() { return this.form.controls; }

    onSubmit() {
        this.submitted = true;

        // reset alerts on submit
        this.alertService.clear();

        // stop here if form is invalid
        if (this.form.invalid) {
            return;
        }

        this.accountService.login(this.forms.username.value, this.forms.password.value)
            //.pipe(first())
            .subscribe({
                next: () => {
```

```
                    // get return url from query parameters or default to home page
                    const returnUrl = this.route.snapshot.queryParams['Url'] ;
                    this.router.navigateByUrl(returnUrl);
                },
                error: error => {
                    this.alertService.error(error);
                }
            });
    }
}
```

app.component.html

```
<nav class="navbar navbar-expand  bg-light px-3" *ngIf="user">
    <div class="navbar-nav">
        <a class="nav-item nav-link" routerLink="/" routerLinkActive="active"
[routerLinkActiveOptions]="{exact: true}">Home</a>
        <a class="nav-item nav-link" routerLink="/users" routerLinkActive="active">Users</a>
        <button class="btn btn-link nav-item nav-link" id="logout" (click)="logout()">Logout</button>
    </div>
</nav>

<div [ngClass]="{ 'bg-light': user }">
    <alert></alert>
    <router-outlet></router-outlet>
</div>
```

app.component.ts

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

import { AccountService } from './_services';
import { User } from './_models';

@Component({ selector: 'app-root', templateUrl: 'app.component.html' })
export class AppComponent {
    user?: User | null;

    constructor(private accountService: AccountService) {
        this.accountService.user.subscribe(x => this.user = x);
    }

    logout() {
        this.accountService.logout();
    }
}
```

account.service.ts

```
import { Injectable } from '@angular/core';
import { Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';
import { BehaviorSubject, Observable } from 'rxjs';
import { map } from 'rxjs/operators';

import { environment } from '@environments/environment';
import { User, admin } from '@app/_models/user';
```

```
@Injectable({ providedIn: 'root' })
export class AccountService {
    private userSubject: BehaviorSubject<User | null>;
    public user: Observable<User | null>;
    ad=new admin("san","san")

    constructor(
        private router: Router,
        private http: HttpClient
    )

    {
        this.userSubject = new BehaviorSubject(JSON.parse(localStorage.getItem('user')!));
        this.user = this.userSubject.asObservable();

    }

    public get userValue() {
        return this.userSubject.value;
    }

    login(username: string, password: string) {

     //if(this.ad.password!==password){}
        console.log(this.ad.password+"logging");
        console.log(password+"Entered");
        return this.http.post<User>(`http://localhost:8080/product/authenticate`, { username, password })
            .pipe(map(user => {
                // store user details and jwt token in local storage to keep user logged in between page
refreshes
                localStorage.setItem('user', JSON.stringify(user));
                this.userSubject.next(user);
                return user;
            }));

    }

    logout() {
        // remove user from local storage and set current user to null
        localStorage.removeItem('user');
        this.userSubject.next(null);
        this.router.navigate(['/account/login']);
    }

    register(user: User) {
        //return this.http.post(`http://localhost:4000/users/register`, user);
        return this.http.post(`http://localhost:8080/product/add`, user);
    }

    getAll() {
        //return this.http.get<User[]>(`${environment.apiUrl}/users`);
        return this.http.get<User[]>(`http://localhost:8080/product/list`);
    }

    getById(id: string) {
        return this.http.get<User>(`http://localhost:8080/product/list/${id}`);
    }

    addProductToBackEnd(name: string, price: String): Observable<User> {
        return this.http.post<User>(`http://localhost:8080/product/add`,
          {
            "name": name,
```

```
            "price": price,
          }
        );
      }


    update(id: string, params: any) {
        return this.http.put(`http://localhost:8080/product/update/${id}`, params)
            .pipe(map(x => {
                // update stored user if the logged in user updated their own record
                if (id == this.userValue?.id) {
                    // update local storage
                    const user = { ...this.userValue, ...params };
                    localStorage.setItem('user', JSON.stringify(user));

                    // publish updated user to subscribers
                    this.userSubject.next(user);
                }
                return x;
            }));
    }

    delete(id: string) {
        return this.http.delete(`http://localhost:8080/product/delete/${id}`)
            .pipe(map(x => {
                // auto logout if the logged in user deleted their own record
                if (id == this.userValue?.id) {
                    this.logout();
                }
                return x;
            }));
    }
}
```

ProductRestController.java

```
package com.ecommerce;

import java.util.List;
import java.util.Optional;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;


@RestController
@RequestMapping("/product")
@CrossOrigin
public class ProductRestController {

        @Autowired
        EProductRepositry eProductRepo;

        // List all the products
        @GetMapping(path="/list", produces = "application/json")
        public List<EProduct> listProducts(){
                List<EProduct> products = eProductRepo.findAll();
```

```
                return products;
        }

        // Adding a new product
        @PostMapping(path="/add", consumes="application/json" , produces = "application/json")
        public EProduct addProduct(@RequestBody EProduct eProduct){
                eProduct = eProductRepo.save(eProduct);
                return eProduct;
        }

        // Finding a single product and fetching its details
        @GetMapping(path="/list/{id}", produces = "application/json")
        public Object showProduct(@PathVariable("id") int id){

                Optional<EProduct> productFromRepo = eProductRepo.findById(id);

                if (productFromRepo.isPresent()) {
                        EProduct product = productFromRepo.get();
                        return product;
                }else {
                        return "Aadhar with id = "+ id + " not found";
                }
        }

        @GetMapping(path="/update/{id}", produces = "application/json")
        public Object updateproduct(@PathVariable("id") int id,@RequestBody EProduct eProduct){

                Optional<EProduct> productFromRepo = eProductRepo.findById(id);
                if (productFromRepo.isPresent()) {
                                eProductRepo.update(eProduct);
                                List<EProduct> products = eProductRepo.findAll();
                                return products;

                }else {
                        return "Aadhar with id = "+ id + " not found, cannot update";
                }
        }
        //Delete a Product
        @GetMapping(path="/delete/{id}", produces = "application/json")
        public Object deleteProduct(@PathVariable("id") int id){

                Optional<EProduct> productFromRepo = eProductRepo.findById(id);

                if (productFromRepo.isPresent()) {
                        eProductRepo.deleteById(id);
                        return "Product with id = "+ id + " found and deleted";
                }else {
                        return "Product with id = "+ id + " not found";
                }
        }

}
```

EProduct.java

```
package com.ecommerce;

import java.math.BigDecimal;
import java.sql.Date;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.NamedQuery;
import jakarta.persistence.Table;
```

```java
@NamedQuery(name ="EProduct.findAllWherePriceIs1000", query="SELECT p from EProduct p where p.price=1000")
@Entity
@Table(name="eproduct")
public class EProduct {


                @Id
                @Column(name="Aadhar_id")
                private long ID;

        private String name;
        private BigDecimal price;


        public EProduct() {
        }

        public long getID() {return this.ID; }
        public String getName() { return this.name;}
        public BigDecimal getPrice() { return this.price;}

        public void setID(long id) { this.ID = id;}
        public void setName(String name) { this.name = name;}
        public void setPrice(BigDecimal price) { this.price = price;}

}
```

account.module.ts

```typescript
import { NgModule } from '@angular/core';
import { ReactiveFormsModule } from '@angular/forms';
import { CommonModule } from '@angular/common';


@NgModule({
    imports: [
        CommonModule,
        ReactiveFormsModule,
    ],

    declarations: [
    ]
})

export class AccountModule { }
```

add-product.component.ts

```typescript
import { Component } from '@angular/core';
import { Product } from '../classes/Product';
import { ProductService } from '../product.service';

@Component({
  selector: 'add-product',
  templateUrl: './add-product.component.html',
  styleUrls: ['./add-product.component.css']
})
export class AddProductComponent {

  product:Product = new Product('',0,'',true,true);
  constructor(private productService: ProductService) {
  }
```

```
  OnSubmit():void{
    console.log(`Form submitted with values ${this.product.name} ,
${this.product.price,this.product.id}`);

this.productService.addProductToBackEnd(this.product.name,this.product.price,this.product.id).subscribe(
      (response) => { this.product = response; },
      (error) => { console.log(error); });
  }

}
```

add-product.component.html

```
<div class="container" >

    <form #addProductForm="ngForm" (ngSubmit)="OnSubmit()">

        <div class="form-group">

            <label for="name">Name</label>

            <input type="text" class="form-control" id="name"  name="name" required
            [(ngModel)]="product.name"     #name="ngModel">

            <div [hidden]="name.valid || name.pristine"       class="alert alert-danger">
            Name is required
            </div>

        </div>

        <div class="form-group">

            <label for="price">Phone number</label>

            <input type="text" class="form-control" id="price"  name="price" required
            [(ngModel)]="product.price"   minlength="2" maxlength="8"            #price="ngModel">

            <div [hidden]="price.valid || price.pristine"       class="alert alert-danger">
                Price should be b/w 2 and 6 digits.
                </div>

        </div>

        <div class="form-group">

            <label for="price">Aadhar ID</label>

            <input type="text" class="form-control" id="iden"  name="i" required
            [(ngModel)]="product.id"   minlength="2" maxlength="9"            #price="ngModel">

            <div [hidden]="price.valid || price.pristine"       class="alert alert-danger">
                Price should be b/w 2 and 6 digits.
                </div>
        </div>

        <button  [disabled]="!name.valid" type="submit" class="btn btn-primary" > Register </button>
    </form>

</div>
```

application.properties

```
spring.datasource.url=jdbc:mysql://localhost:3306/aadhar
spring.datasource.driver-className=com.mysql.cj.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=1234
spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

#spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect

spring.jpa.show-sql=true

logging.level.org.hibernate.SQL=DEBUG
logging.level.org.hibernate.type=TRACE
```