**Capstone Project Source Code**

**APP.COMPONENT.HTML**

<div class="mb-1">

<app-grocery-header (sectionChanged)="onSectionChanged($event)"></app-grocery-header>

<router-outlet></router-outlet>

</div>

**APP.COMPONENT.TS**

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

import { AuthService } from './services/auth.service';

@Component({

selector: 'app-root',

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

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

})

export class AppComponent implements OnInit {

constructor(private authService:AuthService){}

title = 'GroceryApp';

ngOnInit(): void {

this.authService.autoAuthUser();

}

onSectionChanged(event:any){

}

}

**LOGIN.COMPONENT.CSS**

.login\_container{

position: absolute;

top: 50%; right: 50%;

transform: translate(50%,-50%);

background: blue;

padding: 2em;

}

.container-fluid{

position: absolute;

height:100%;

width:100%;

background:linear-gradient( to left top, blue, red);

}

**LOGIN.COMPONENT.HTML**

<div class="container-fluid">

<form [formGroup]="loginForm">

<div class="login\_container">

<h4 class="text-center mb-2">Grocery Login</h4>

<div class="input-group mb-3">

<div class="input-group-text">

<i class="fa fa-user" aria-hidden="true"></i>

</div>

<input type="email" class="form-control" placeholder="Email" formControlName="email" aria-label="Text input with checkbox">

</div>

<div class="input-group mb-3">

<div class="input-group-text">

<i class="fa fa-key" aria-hidden="true"></i>

</div>

<input type="password" class="form-control" formControlName="password" placeholder="Password"

aria-label="Text input with radio button">

</div>

<div class="text-warning mb-2" \*ngIf="isLoginClicked && !loginService.loggedIn">

Incorrect username or password

</div>

<div class="form-group text-center" style="display: flex;justify-content: center; align-items: center;">

<button type="button" class="btn btn-primary mx-2" (click)="loginUser($event)">Login</button>

<button type="button" class="btn btn-primary mx-2" (click)="signUp($event)">SignUp</button>

</div>

</div>

</form>

<p-progressSpinner \*ngIf="onLoading"></p-progressSpinner>

</div>

**LOGIN.COMPONENT.TS**

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

import { FormControl, FormGroup, Validators } from '@angular/forms';

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

import { AuthService } from 'src/app/services/auth.service';

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-login',

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

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

})

export class LoginComponent {

onLoading:boolean = false;

email: any;

password: any;

isLoginClicked: boolean = false;

loginForm = new FormGroup({

email: new FormControl('', Validators.required),

password: new FormControl('', Validators.required),

});

constructor(private router: Router, public loginService: LoginApiService, public authService:AuthService) { }

loginUser(event: any) {

if(this.loginForm.invalid){

return;

}

this.email = this.loginForm.value.email;

this.password = this.loginForm.value.password;

this.onLoading = true;

this.authService.loginUser(this.email, this.password);

}

signUp(event: any) {

if(this.loginForm.invalid){

return;

}

this.email = this.loginForm.value.email;

this.password = this.loginForm.value.password;

this.authService.createUser(this.email, this.password);

}

ngOnInit(): void {

}

}

**HOME-GROCERY.COMPONENT.HTML**

<div class="container-fluid">

<app-product-section [products]="products" [carouselDetails]="carouselDetails"></app-product-section>

</div>

**HOME-GROCERY.COMPONENT.CSS**

.flex-container {

margin: 2% 2%;

display: flex;

flex-wrap: wrap;

flex-direction: row;

gap: 2rem;

}

**HOME-GROCERY.COMPONENT.TS**

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

import { Subscription } from 'rxjs';

import { Name } from 'src/app/models/api-models';

import { ApiService } from 'src/app/services/api.service';

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-home-grocery',

templateUrl: './home-grocery.component.html',

styleUrls: ['./home-grocery.component.css']

})

export class HomeGroceryComponent implements OnInit {

carouselDetails:any;

products:any;

productsSub:Subscription;

productName:Name;

category:string;

productCategory:any;

name:string;

constructor(public api: LoginApiService, private service:ApiService) { }

ngOnInit(): void {

this.carouselDetails = this.api.carouselDetails;

this.service.getProducts();

this.productsSub = this.service.getPostUpdateListener()

.subscribe((data:any)=>{

this.products = data;

this.api.allPages = this.products;

this.products.forEach((item:any)=>{

item.isWishlist=false;

});

})

}

}

**PRODUCT-SECTION.COMPONENT.HTML**

<div>

<app-carousel-section [carouselDetails]="carouselDetails"></app-carousel-section>

</div>

<div class="flex-container">

<div \*ngFor="let item of products; let i=index;">

<app-product [product]="item" (productAddedToCart)="onNewProductsAdded($event)" (productWishlisted)="onProductWishlisted($event)"></app-product>

</div>

</div>

<div class="row" style="display:flex; justify-content: right; align-items: right;position:sticky;bottom:0;">

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:10rem;" (click)="onCart($event)">Cart</button>

<button type="button" class="btn btn-secondary" style="border:1px solid black;margin:1%; width:10rem;" [disabled]="!wishlistPresent" (click)="onWishlist($event)">Wishlist</button>

<button type="button" class="btn btn-danger" style="border:1px solid black;margin:1%; width:10rem;" (click)="onCheckOut($event)">CheckOut</button>

</div>

<div \*ngIf="dialog.visibility">

<p-dialog [(visible)]="dialog.visibility" position="center">

<ng-template pTemplate="header">

<h5 style="font-size:16px;font-weight:700;color:black">{{dialog.title}}</h5>

</ng-template>

<ng-template pTemplate="content">

<p-table [value]="cartProduct" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>ProductName</th>

<th>Category</th>

<th>Quantity</th>

<th>Price</th>

<th>Total Price</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-product>

<tr>

<td>{{product.productName}}</td>

<td>{{product.category}}</td>

<td>{{product.quantity}}</td>

<td>{{product.price}}</td>

<td>{{product.price \* product.quantity}}</td>

</tr>

</ng-template>

<ng-template pTemplate="footer">

</ng-template>

</p-table>

</ng-template>

<ng-template pTemplate="footer">

<div class="row" style="display:flex;justify-content: center;align-items: center;">

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:10rem;" (click)="onCloseCart($event)">Close Cart</button>

</div>

</ng-template>

</p-dialog>

</div>

**PRODUCT-SECTION.COMPONENT.CSS**

.flex-container {

margin: 2% 2%;

display: flex;

flex-wrap: wrap;

flex-direction: row;

gap: 2rem;

}

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

**PRODUCT-SECTION.COMPONENT.TS**

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

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

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-product-section',

templateUrl: './product-section.component.html',

styleUrls: ['./product-section.component.css']

})

export class ProductSectionComponent implements OnInit {

@Input() products:any;

@Input() carouselDetails:any;

wishlistPresent:boolean=false;

amount:any=0;

cartProduct:any;

dialog:any={message:'', title:'', visibility:false};

constructor(public api:LoginApiService, private router:Router){}

ngOnInit(): void {

console.log(this.products);

}

onCheckOut(event:any){

this.router.navigate(['cart']);

}

onNewProductsAdded(event:any){

console.log(this.api.cartProduct);

}

onProductWishlisted(event:any){

if(event.length > 0){

this.wishlistPresent = true;

}else{

this.wishlistPresent = false;

}

}

onCart(event:any){

this.cartProduct = this.api.cartProduct;

this.dialog.title = 'Cart details';

this.dialog.visibility = true;

}

onCloseCart(event:any){

this.dialog.visibility = false;

}

onWishlist(event:any){

this.router.navigate(['wishlist']);

}

}

**CAROUSEL-SECTION.COMPONENT.HTML**

<div id="carouselExampleCaptions" class="carousel slide">

<div class="carousel-indicators">

<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" class="active"

aria-current="true" aria-label="Slide 1"></button>

<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1"

aria-label="Slide 2"></button>

<button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2"

aria-label="Slide 3"></button>

</div>

<div class="carousel-inner">

<div class="carousel-item active">

<img [src]="carouselDetails[0].image" class="d-block w-100" alt="Biscuits" height="500">

<div class="carousel-caption d-none d-md-block">

<h5>{{carouselDetails[0].captionHeader}}</h5>

<p>{{carouselDetails[0].captionDescription}}</p>

</div>

</div>

<div class="carousel-item">

<img [src]="carouselDetails[1].image" class="d-block w-100" alt="Fruits" height="500">

<div class="carousel-caption d-none d-md-block">

<h5>{{carouselDetails[1].captionHeader}}</h5>

<p>{{carouselDetails[1].captionDescription}}</p>

</div>

</div>

<div class="carousel-item">

<img [src]="carouselDetails[2].image" class="d-block w-100" alt="Vegetables" height="500">

<div class="carousel-caption d-none d-md-block">

<h5>{{carouselDetails[2].captionHeader}}</h5>

<p>{{carouselDetails[2].captionDescription}}</p>

</div>

</div>

</div>

<button class="carousel-control-prev" type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide="prev">

<span class="carousel-control-prev-icon" aria-hidden="true"></span>

<span class="visually-hidden">Previous</span>

</button>

<button class="carousel-control-next" type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide="next">

<span class="carousel-control-next-icon" aria-hidden="true"></span>

<span class="visually-hidden">Next</span>

</button>

</div>

**CAROUSEL-SECTION.COMPONENT.TS**

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

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-carousel-section',

templateUrl: './carousel-section.component.html',

styleUrls: ['./carousel-section.component.css']

})

export class CarouselSectionComponent implements OnInit {

@Input()carouselDetails:any;

constructor(public api:LoginApiService){}

ngOnInit(): void {

}

}

**CART-VIEW.COMPONENT.HTML**

<div class="container-fluid">

<h5 style="font-weight: 700;color:black;font-size:22px;" class="mb-3">Cart Section</h5>

<p-table [value]="products" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>ProductName</th>

<th>Category</th>

<th>Quantity</th>

<th>Price</th>

<th>Total Price</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-product>

<tr>

<td>{{product.productName}}</td>

<td>{{product.category}}</td>

<td>{{product.quantity}}</td>

<td>{{product.price}}</td>

<td>{{product.price \* product.quantity}}</td>

</tr>

</ng-template>

<ng-template pTemplate="footer">

<tr>

<td>Amount</td>

<td></td>

<td></td>

<td></td>

<td>{{amount}}</td>

</tr>

<tr>

<td>

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:6rem;" (click)="onCancel($event)">Cancel</button>

</td>

<td></td>

<td></td>

<td></td>

<td>

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:6rem;" (click)="onProceed($event)">Proceed</button>

</td>

</tr>

</ng-template>

</p-table>

</div>

<div \*ngIf="dialog.visibility">

<p-dialog [(visible)]="dialog.visibility" position="center">

<ng-template pTemplate="header">

<h5 style="font-size:16px;font-weight:700;color:black">{{dialog.title}}</h5>

</ng-template>

<ng-template pTemplate="content">

<p style="font-size:14px; font-weight:600;color:black;">{{dialog.message}}</p>

</ng-template>

<ng-template pTemplate="footer">

<div style="display:flex;justify-content: center;align-items: center;" \*ngIf="dialog.title === 'Order Confirmation'">

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:6rem;" (click)="onOrderNo($event)">No</button>

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:6rem;" (click)="onOrderYes($event)">Yes</button>

</div>

<div style="display:flex;justify-content: center;align-items: center;" \*ngIf="dialog.title === 'Confirmed'">

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:6rem;" (click)="onOrderConfirmation($event)">Yes</button>

</div>

</ng-template>

</p-dialog>

</div>

**CART-VIEW.COMPONENT.TS**

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

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

import { ApiService } from 'src/app/services/api.service';

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-cart-view',

templateUrl: './cart-view.component.html',

styleUrls: ['./cart-view.component.css']

})

export class CartViewComponent implements OnInit {

constructor(private router:Router, private api:LoginApiService, private service:ApiService){}

ngOnInit(): void {

this.api.cartProduct.forEach((item:any)=>{

this.amount += (item.price \* item.quantity);

console.log(this.amount);

});

}

dialog:any={message:'',title:'', visibility:false};

products:any=this.api.cartProduct;

amount:any=0;

onCancel(event:any){}

onProceed(event:any){

this.dialog.title = 'Order Confirmation';

this.dialog.message = 'Do you wish to place your order?'

this.dialog.visibility = true;

}

onOrderYes(event:any){

this.dialog.visibility = false;

console.log(this.api.cartProduct);

this.service.createOrder(this.api.cartProduct);

this.service.getProductOrderUpdatedListener()

.subscribe((data:any)=>{

this.dialog.title = 'Confirmed';

this.dialog.message = `Your order is confirmed. Your order no: is ${data.orderId}`;

this.dialog.visibility = true;

})

}

onOrderNo(event:any){

this.dialog.visibility = false;

}

onOrderConfirmation(event:any){

this.dialog.visibility = false;

this.router.navigate(['home']);

}

}

**CART-VIEW.COMPONENT.CSS**

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

:host ::ng-deep .p-datatable .p-datatable-tfoot > tr > td {

background:grey;

}

**GROCERY-HEADER.COMPONENT.TS**

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

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

import { Subscription } from 'rxjs';

import { AuthService } from 'src/app/services/auth.service';

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-grocery-header',

templateUrl: './grocery-header.component.html',

styleUrls: ['./grocery-header.component.css']

})

export class GroceryHeaderComponent implements OnInit, OnDestroy {

onLoading:boolean = false;

userIsAuthenticated:boolean = false;

isAdmin:boolean = false;

authListenerSubs:Subscription;

adminListener:Subscription;

@Output () sectionChanged = new EventEmitter<any>();

constructor(private router:Router, public api:LoginApiService, private authService:AuthService){}

ngOnDestroy(): void {

this.authListenerSubs.unsubscribe();

this.adminListener.unsubscribe();

}

ngOnInit(): void {

this.isAdmin = this.authService.getIsAdmin();

this.userIsAuthenticated = this.authService.getIsAuth();

this.authListenerSubs = this.authService.getAuthStatusListener()

.subscribe(isAuthenticated=>{

this.userIsAuthenticated = isAuthenticated;

});

this.adminListener = this.authService.getAdminListener()

.subscribe(admin=>{

this.isAdmin = admin;

});

}

onClickSection(event:any, section:any){

debugger

console.log(section)

this.api.selectedSection = section;

this.api.allPages.forEach((item:any, index:any)=>{

if(item.category === this.api.selectedSection){

this.api.productPage = this.api.allPages[index];

console.log(this.api.productPage)

}

})

this.sectionChanged.emit(section);

this.router.navigate(['section']) ;

}

onHome(event:any){

this.router.navigate(['home']);

}

onAdd(event:any){

this.router.navigate(['add']);

}

onView(event:any){

this.router.navigate(['view']);

}

onUsers(event:any){

this.router.navigate(['user-view']);

}

onOrders(event:any){

this.router.navigate(['order']);

}

onLogout(event:any){

this.onLoading = true;

this.authService.logout();

}

onLogin(event:any){

this.router.navigate(['login']);

}

}

**GROCERY-HEADER.COMPONENT.HTML**

<nav class="navbar navbar-expand-lg bg-body-tertiary bg-primary" data-bs-theme="dark">

<div class="container-fluid">

<a class="navbar-brand me-4" href="#">Big Groceries</a>

<button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"

aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="collapse navbar-collapse" id="navbarSupportedContent">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item p-2" \*ngIf="userIsAuthenticated">

<a class="nav-link" aria-current="page" (click)="onHome($event)">Home</a>

</li>

<li class="nav-item p-2" \*ngIf="userIsAuthenticated && isAdmin">

<a class="nav-link" aria-current="page" (click)="onAdd($event)">Add Product</a>

</li>

<li class="nav-item p-2"\*ngIf="userIsAuthenticated && isAdmin">

<a class="nav-link" aria-current="page" (click)="onView($event)">View Product</a>

</li>

<li class="nav-item p-2" \*ngIf="userIsAuthenticated && isAdmin">

<a class="nav-link" aria-current="page" (click)="onUsers($event)">Users</a>

</li>

</ul>

<span class="spacer"></span>

<form class="d-flex" role="search">

<input class="form-control me-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-light" type="submit">Search</button>

</form>

<div style="float:right">

<ul class="navbar-nav me-auto mb-2 mb-lg-0">

<li class="nav-item p-2" \*ngIf="!userIsAuthenticated">

<a class="nav-link active" aria-current="page" (click)="onLogin($event)">Login</a>

</li>

<li class="nav-item dropdown p-2" \*ngIf="userIsAuthenticated">

<a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown"

aria-expanded="false">

User

</a>

<ul class="dropdown-menu">

<li (click)="onOrders($event)"><a class="dropdown-item">

My Order</a></li>

</ul>

</li>

<li class="nav-item p-2" \*ngIf="userIsAuthenticated">

<a class="nav-link" aria-current="page" (click)="onLogout($event)">Logout</a>

</li>

<li class="nav-item p-2" \*ngIf="!userIsAuthenticated">

<a class="nav-link" aria-current="page">SignUp</a>

</li>

</ul>

</div>

</div>

</div>

</nav>

<!-- <p-progressSpinner \*ngIf="onLoading"></p-progressSpinner> -->

**GROCERY-HEADER.COMPONENT.CSS**

.nav-link,.navbar-brand {

color: white;

}

.spacer{

flex:1 1 auto;

}

**WISHLIST-ITEMS.COMPONENT.HTML**

<div class="container-fluid">

<div class="row" class="mb-3" style="display:flex; justify-content: right; align-items: right;position:sticky;top:5%">

<button type="button" class="btn btn-danger" style="border:1px solid black;margin:1%; width:10rem;" (click)="onHome($event)">Home</button>

</div>

<p-table [value]="wishlist" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>ProductName</th>

<th>Image</th>

<th>Category</th>

<th>Price</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-product>

<tr>

<td>{{product.productName}}</td>

<td><img [src]="product.image" class="d-block w-100" alt="Biscuits" height="100" width="100"/></td>

<td>{{product.category}}</td>

<td>{{product.price}}</td>

</tr>

</ng-template>

</p-table>

</div>

**WISHLIST-ITEMS.COMPONENT.TS**

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

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

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-wishlist-items',

templateUrl: './wishlist-items.component.html',

styleUrls: ['./wishlist-items.component.css']

})

export class WishlistItemsComponent implements OnInit {

wishlist:any;

constructor(private api:LoginApiService, private router:Router){}

ngOnInit(): void {

this.wishlist = this.api.wishProduct;

}

onHome(event:any){

this.router.navigate(['home']);

}

}

**WISHLIST-ITEMS.COMPONENT.CSS**

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

:host ::ng-deep .p-datatable .p-datatable-tfoot > tr > td {

background:grey;

}

**ORDER-VIEW.COMPONENT.HTML**

<div class="container-fluid">

<h5 style="font-weight: 700;color:black;font-size:22px;" class="mb-3">Order Section</h5>

<div \*ngFor="let item of orders">

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:15rem;" (click)="displayItems($event, item.\_id)">{{item.\_id}}</button>

</div>

<div \*ngIf="isTableVisible">

<p-table [value]="selectedOrder" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>ProductName</th>

<th>Category</th>

<th>Quantity</th>

<th>Price</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-order>

<tr>

<td>{{order.productName}}</td>

<td>{{order.category}}</td>

<td>{{order.quantity}}</td>

<td>{{order.price}}</td>

</tr>

</ng-template>

<ng-template pTemplate="footer">

<tr>

<td>Amount</td>

<td></td>

<td></td>

<td>{{amount}}</td>

</tr>

</ng-template>

</p-table>

</div>

</div>

**ORDER-VIEW.COMPONENT.TS**

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

import { Subscription } from 'rxjs';

import { ApiService } from 'src/app/services/api.service';

@Component({

selector: 'app-order-view',

templateUrl: './order-view.component.html',

styleUrls: ['./order-view.component.css']

})

export class OrderViewComponent implements OnInit, OnDestroy {

constructor(private service:ApiService){}

ngOnDestroy(): void {

this.orderSub.unsubscribe();

}

orders:any;

orderSub:Subscription;

amount:any=0;

index:any;

isTableVisible:boolean=false;

selectedOrder:any;

onCancel(event:any){}

displayItems(event:any,id:any){

this.index = this.orders.find((x:any)=>x.\_id === id);

this.selectedOrder = this.index.product;

this.selectedOrder.forEach((item:any)=>{

this.amount += (item.price \* item.quantity);

});

this.isTableVisible = true;

}

ngOnInit(): void {

this.service.getOrder();

this.orderSub = this.service.getProductOrderUpdatedListener()

.subscribe(res=>{

this.orders = res.order;

console.log(this.orders);

});

}

}

**ORDER-VIEW.COMPONENT.CSS**

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

:host ::ng-deep .p-datatable .p-datatable-tfoot > tr > td {

background:grey;

}

**PRODUCT.COMPONENT.TS**

import { Component, EventEmitter, Input, Output } from '@angular/core';

import { LoginApiService } from 'src/app/services/login-api.service';

@Component({

selector: 'app-product',

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

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

})

export class ProductComponent {

constructor(public api: LoginApiService) { }

@Output() productAddedToCart = new EventEmitter<any>();

@Output() productWishlisted = new EventEmitter<any>();

@Input() product: any;

quantity:any='';

onAddToCart(event: any) {

if (this.api.cartProduct.length === 0) {

this.product.quantity = this.quantity;

this.api.cartProduct.push(this.product);

} else {

let index = this.api.cartProduct.findIndex((record: any) => record.id === this.product.id);

if (index > -1) {

this.api.cartProduct.splice(index, 1);

}

this.product.quantity = this.quantity;

this.api.cartProduct.push(this.product);

this.productAddedToCart.emit(true);

}

}

isWishlistClicked(event: any) {

debugger

this.product.isWishlist = !this.product.isWishlist;

if(this.product.isWishlist)

{

this.api.wishProduct.push(this.product);

}else

{

let wishlistedProductPresent = this.api.wishProduct.filter((x:any)=>x.productName === this.product.productName);

if(wishlistedProductPresent){

let index = this.api.wishProduct.findIndex((x:any)=>x.productName === this.product.productName);

this.api.wishProduct.splice(index, 1);

}

}

this.productWishlisted.emit(this.api.wishProduct);

// this.api.productDetails.forEach((item: any, index: any) => {

// if (item.id === this.product.id && item.isWishlist === false) {

// this.api.productDetails[index].isWishlist = true;

// } else if (item.id === this.product.id && item.isWishlist === true) {

// this.api.productDetails[index].isWishlist = false;

// }

// });

// let wishlistProducts: any = [];

// this.api.productDetails.forEach((item: any) => {

// if (item.isWishlist === true) {

// wishlistProducts.push(item);

// }

// });

// this.api.wishProduct = wishlistProducts;

}

}

**PRODUCT.COMPONENT.HTML**

<div class="card border-primary mb-3 me-2" style="width: 20rem;">

<div class="card-header">

<div class="row">

<div class="col-12">

<span style="float:right" \*ngIf="!product.isWishlist" (click)="isWishlistClicked($event)">

<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-heart"

viewBox="0 0 16 16">

<path

d="m8 2.748-.717-.737C5.6.281 2.514.878 1.4 3.053c-.523 1.023-.641 2.5.314 4.385.92 1.815 2.834 3.989 6.286 6.357 3.452-2.368 5.365-4.542 6.286-6.357.955-1.886.838-3.362.314-4.385C13.486.878 10.4.28 8.717 2.01L8 2.748zM8 15C-7.333 4.868 3.279-3.04 7.824 1.143c.06.055.119.112.176.171a3.12 3.12 0 0 1 .176-.17C12.72-3.042 23.333 4.867 8 15z" />

</svg>

</span>

<span style="float:right" \*ngIf="product.isWishlist" (click)="isWishlistClicked($event)">

<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" class="bi bi-heart-fill"

viewBox="0 0 16 16">

<path fill-rule="evenodd"

d="M8 1.314C12.438-3.248 23.534 4.735 8 15-7.534 4.736 3.562-3.248 8 1.314z" />

</svg>

</span>

</div>

</div>

<div class="row">

<div class="col-12">

<span class="mx-2" style="text-align:center">

<h5 class="card-title">{{product.productName}}</h5>

</span>

</div>

</div>

<img [src]="product.image" class="d-block w-100" alt="Biscuits" height="200" width="200"/>

</div>

<div class="card-body text-primary">

<div class="row">

<div class="col-8">

<label for="quantity" style="font-weight:600;color:lightgrey;font-size:14px">Price</label>

<p style="font-weight:700;font-size: 22px; color:rgba(0,3,3,0.9)">{{product.price}}</p>

</div>

<div class="col-4" style="text-align: right;">

<label for="quantity" style="font-weight:600;color:lightgrey;font-size:14px">Quantity</label>

<p-inputNumber [(ngModel)]="quantity" mode="decimal" [showButtons]="true"

buttonLayout="horizontal" spinnerMode="horizontal" decrementButtonClass="p-button-secondary"

incrementButtonClass="p-button-secondary" incrementButtonIcon="pi pi-plus"

decrementButtonIcon="pi pi-minus"></p-inputNumber>

</div>

</div>

</div>

<div class="card-footer">

<span style="display: flex;justify-content: center;align-items: center;"

(click)="onAddToCart($event)">

<button type="button" class="btn btn-primary"><svg xmlns="http://www.w3.org/2000/svg" width="16" height="16"

fill="currentColor" class="bi bi-bag-check" viewBox="0 0 16 16">

<path fill-rule="evenodd"

d="M10.854 8.146a.5.5 0 0 1 0 .708l-3 3a.5.5 0 0 1-.708 0l-1.5-1.5a.5.5 0 0 1 .708-.708L7.5 10.793l2.646-2.647a.5.5 0 0 1 .708 0z" />

<path

d="M8 1a2.5 2.5 0 0 1 2.5 2.5V4h-5v-.5A2.5 2.5 0 0 1 8 1zm3.5 3v-.5a3.5 3.5 0 1 0-7 0V4H1v10a2 2 0 0 0 2 2h10a2 2 0 0 0 2-2V4h-3.5zM2 5h12v9a1 1 0 0 1-1 1H3a1 1 0 0 1-1-1V5z" />

</svg><span class="mx-2 my-2">Add to Bag</span></button>

</span>

</div>

</div>

**PRODUCT.COMPONENT.CSS**

:host ::ng-deep .p-inputnumber-buttons-horizontal .p-inputnumber-input{

width: 200%!important;

}

:host ::ng-deep .p-inputnumber-buttons-horizontal .p-button.p-inputnumber-button-up{

width: 25%!important;

}

:host ::ng-deep .p-inputnumber-buttons-horizontal .p-button.p-inputnumber-button-down{

width: 25%!important;

}

**PRODUCT-VIEW.COMPONENT.TS**

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

import { Subscription } from 'rxjs';

import { Products } from 'src/app/models/api-models';

import { ApiService } from 'src/app/services/api.service';

@Component({

selector: 'app-product-view',

templateUrl: './product-view.component.html',

styleUrls: ['./product-view.component.css']

})

export class ProductViewComponent implements OnInit, OnDestroy {

verifyAdmin:boolean = false;

warningDialog:any={message:'You have no access to view this content. Please contact Administrator.', title:'Warning', visibility:false};

productId:any;

editProduct:any;

products:any=[];

productSub:Subscription;

dialog:any={visibility:false,title:''};

categories: any = [

{

id: 1,

name: 'Fruits'

},

{

id: 2,

name: 'Vegetables'

},

{

id: 3,

name: 'Bakery'

}

]

constructor(public api:ApiService){}

ngOnInit(): void {

let userType = localStorage.getItem("userType");

if(userType === 'admin'){

this.verifyAdmin = true;

}else{

this.verifyAdmin = false;

}

if(this.verifyAdmin){

this.api.getProducts();

this.productSub = this.api.getPostUpdateListener()

.subscribe((product:Products[])=>{

this.products = product;

});

}else{

this.warningDialog.visibility = true;

}

}

onEdit(event:any, product:any){

this.productId = product.id;

const categories = this.categories.filter((x:any)=>x.name === product.category );

let formattedProduct={

productName:product.productName,

price:product.price,

category:categories[0]

}

this.editProduct = formattedProduct;

console.log(formattedProduct);

this.dialog.title = 'Edit Product';

this.dialog.visibility=true;

}

onDelete(event:any,product:any){

this.api.deleteProducts(product.id);

}

onUpdatedProduct(event:any){

this.dialog.visibility = false;

}

ngOnDestroy(): void {

if(this.verifyAdmin){

this.productSub.unsubscribe();

}

}

}

**PRODUCT-VIEW.COMPONENT.HTML**

<div class="container-fluid" \*ngIf="verifyAdmin">

<p-table [value]="products" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>ProductName</th>

<th>Category</th>

<th>Price</th>

<th>Actions</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-product>

<tr>

<td>{{product.productName}}</td>

<td>{{product.category}}</td>

<td>{{product.price}}</td>

<td>

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:10rem;" (click)="onEdit($event,product)" >Edit</button>

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:10rem;" (click)="onDelete($event,product)" >Delete</button>

</td>

</tr>

</ng-template>

</p-table>

</div>

<div \*ngIf="dialog.visibility">

<p-dialog [(visible)]="dialog.visibility" position="center">

<ng-template pTemplate="header">

<h5 style="font-size:16px;font-weight:700;color:black">{{dialog.title}}</h5>

</ng-template>

<ng-template pTemplate="content">

<app-edit-product [product]="editProduct" [id]="productId" (productUpdated)="onUpdatedProduct($event)"></app-edit-product>

</ng-template>

<ng-template pTemplate="footer">

</ng-template>

</p-dialog>

</div>

<app-acess-alert [dialog]="warningDialog"></app-acess-alert>

**PRODUCT-VIEW.COMPONENT.CSS**

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

**USER-VIEW.COMPONENT.TS**

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

import { Subscription } from 'rxjs';

import { AuthService } from 'src/app/services/auth.service';

@Component({

selector: 'app-user-view',

templateUrl: './user-view.component.html',

styleUrls: ['./user-view.component.css']

})

export class UserViewComponent implements OnInit, OnDestroy {

user: any;

userSub: Subscription;

verifyAdmin: boolean = false;

dialog:any={message:'You have no access to view this content. Please contact Administrator.', title:'Warning', visibility:false};

constructor(private authService: AuthService) { }

ngOnDestroy(): void {

if(this.verifyAdmin){

this.userSub.unsubscribe();

}

}

ngOnInit(): void {

let userType = localStorage.getItem("userType");

if(userType === 'admin'){

this.verifyAdmin = true;

}else{

this.verifyAdmin = false;

}

if (this.verifyAdmin) {

this.authService.getUserDetails();

this.userSub = this.authService.getUserListener().subscribe((res: any) => {

this.user = res;

});

}else{

this.dialog.visibility = true;

}

}

onDelete(event: any, user: any) {

this.authService.deleteUser(user.\_id);

}

}

**USER-VIEW.COMPONENT.HTML**

<div class="container-fluid" \*ngIf="verifyAdmin">

<p-table [value]="user" [tableStyle]="{'max-width': '60rem'}">

<ng-template pTemplate="header">

<tr>

<th>User</th>

<th>Type</th>

<th>Actions</th>

</tr>

</ng-template>

<ng-template pTemplate="body" let-user>

<tr>

<td>{{user.email}}</td>

<td>{{user.userType}}</td>

<td>

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:10rem;" (click)="onDelete($event,user)" >Delete</button>

</td>

</tr>

</ng-template>

</p-table>

</div>

<app-acess-alert [dialog]="dialog"></app-acess-alert>

**USER-VIEW.COMPONENT.CSS**

:host ::ng-deep .p-datatable .p-datatable-thead > tr > th {

background:blue;

}

:host ::ng-deep .p-datatable .p-datatable-tbody > tr > td{

border-width: 1px;

}

:host ::ng-deep .p-datatable .p-datatable-tfoot > tr > td {

background:grey;

}

**EDIT-PRODUCT.COMPONENT.TS**

import { Component,Input, OnChanges, OnInit,Output,EventEmitter, SimpleChanges } from '@angular/core';

import { FormControl, FormGroup } from '@angular/forms';

import { ApiService } from 'src/app/services/api.service';

@Component({

selector: 'app-edit-product',

templateUrl: './edit-product.component.html',

styleUrls: ['./edit-product.component.css']

})

export class EditProductComponent implements OnInit, OnChanges {

@Input() id:any;

@Input() product:any;

@Output() productUpdated = new EventEmitter<any>();

constructor(public api:ApiService){}

ngOnChanges(changes: SimpleChanges): void {

debugger

// this.editForm.setValue(this.product);

}

ngOnInit(): void {

this.editForm = new FormGroup({

productName: new FormControl(''),

price: new FormControl(''),

category: new FormControl('')

// productImage:new FormControl('')

})

this.editForm.setValue(this.product);

}

editForm: FormGroup;

categories: any = [

{

id: 1,

name: 'Fruits'

},

{

id: 2,

name: 'Vegetables'

},

{

id: 3,

name: 'Bakery'

}

]

changeCategory(event:any){

}

changeImage(event:any){}

onCancel(event:any){

this.editForm.reset();

}

onSave(event:any){

this.api.updateProducts(this.id,this.editForm.value.productName, this.editForm.value.category.name, this.editForm.value.price);

this.editForm.reset();

this.productUpdated.emit(true);

}

}

**EDIT-PRODUCT.COMPONENT.HTML**

<div class="card-section">

<div class="card" style="width:40rem">

<div class="card-body">

<form [formGroup]="editForm">

<div class="row my-1">

<div class="col-6">

<label for="ProductName" style="font-weight: 600;font-size: 14px;color:lightgrey">Product

Name</label>

</div>

<div class="col-6">

<input type="text" pInputText formControlName="productName" />

</div>

</div>

<div class="row my-1">

<div class="col-6">

<label for="ProductName" style="font-weight: 600;font-size: 14px;color:lightgrey">Price</label>

</div>

<div class="col-6">

<input type="text" pInputText formControlName="price" />

</div>

</div>

<div class="row my-1">

<div class="col-6">

<label for="ProductName"

style="font-weight: 600;font-size: 14px;color:lightgrey">Category</label>

</div>

<div class="col-6">

<p-dropdown [options]="categories" formControlName="category" placeholder="Please Select"

optionLabel="name" (onChange)="changeCategory($event)"></p-dropdown>

</div>

</div>

<!-- <div class="row my-1">

<div class="col-6">

<label for="ProductImage" style="font-weight: 600;font-size: 14px;color:lightgrey">Product

Image</label>

</div>

<div class="col-6">

<input type="file" pInputText formControlName="productImage" (onChange)="changeImage($event)"/>

</div>

</div> -->

</form>

</div>

<div class="card-footer" style="display:flex;justify-content: center;align-items: center;">

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:10rem;" (click)="onCancel($event)" >Cancel</button>

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:10rem;" (click)="onSave($event)" >Save</button>

</div>

</div>

</div>

**EDIT-PRODUCT.COMPONENT.CSS**

:host ::ng-deep .p-inputtext{

width:80%!important;

}

:host ::ng-deep .p-dropdown{

width:80%!important;

}

**ADD-PRODUCT.COMPONENT.TS**

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

import { FormControl, FormGroup, Validators } from '@angular/forms';

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

import { ApiService } from 'src/app/services/api.service';

@Component({

selector: 'app-add-product',

templateUrl: './add-product.component.html',

styleUrls: ['./add-product.component.css']

})

export class AddProductComponent implements OnInit {

verifyAdmin:boolean = false;

dialog:any={message:'You have no access to view this content. Please contact Administrator.', title:'Warning', visibility:false};

imagePreview: string = '';

fileList?: FileList | null;

currentFile?: File | null;

constructor(public api: ApiService) { }

ngOnInit(): void {

let userType = localStorage.getItem("userType");

if(userType === 'admin'){

this.verifyAdmin = true;

}else{

this.verifyAdmin = false;

}

if(this.verifyAdmin){

this.productForm = new FormGroup({

productName: new FormControl(''),

price: new FormControl(''),

category: new FormControl('')

});

}else{

this.dialog.visibility = true;

}

}

productForm: FormGroup;

categories: any = [

{

id: 1,

name: 'Fruits'

},

{

id: 2,

name: 'Vegetables'

},

{

id: 3,

name: 'Bakery'

}

]

changeCategory(event: any) {

}

changeImage(event: any) {

this.fileList = event.target.files;

this.currentFile = this.fileList?.item(0);

if (this.fileList) {

const file: File | null = this.fileList.item(0);

if (file) {

this.imagePreview = '';

this.currentFile = file;

this.productForm.value.productImage = this.currentFile;

const reader = new FileReader();

reader.onload = (e: any) => {

this.imagePreview = e.target.result;

};

reader.readAsDataURL(this.currentFile);

}

}

}

onCancel(event: any) {

this.productForm.reset();

}

onSave(event: any) {

this.api.addProducts(this.productForm.value.productName, this.productForm.value.category.name, this.productForm.value.price);

this.productForm.reset();

}

}

**ADD-PRODUCT.COMPONENT.HTML**

<div class="container-fluid" \*ngIf="verifyAdmin">

<div class="card-section" style="position: relative; left:50%; top:50%; transform:translate(-20%,50%)">

<div class="card" style="width:40rem">

<div class="card-header" style="display:flex;justify-content: center;align-items: center;">

<h5 style="font-weight: 700;font-size: 22px;color:black">Add New Product</h5>

</div>

<div class="card-body">

<form [formGroup]="productForm">

<div class="row my-1">

<div class="col-6">

<label for="ProductName" style="font-weight: 600;font-size: 14px;color:lightgrey">Product

Name</label>

</div>

<div class="col-6">

<input type="text" pInputText formControlName="productName" />

</div>

</div>

<div class="row my-1">

<div class="col-6">

<label for="ProductName" style="font-weight: 600;font-size: 14px;color:lightgrey">Price</label>

</div>

<div class="col-6">

<input type="text" pInputText formControlName="price" />

</div>

</div>

<div class="row my-1">

<div class="col-6">

<label for="ProductName"

style="font-weight: 600;font-size: 14px;color:lightgrey">Category</label>

</div>

<div class="col-6">

<p-dropdown [options]="categories" formControlName="category" placeholder="Please Select"

optionLabel="name" (onChange)="changeCategory($event)"></p-dropdown>

</div>

</div>

<!-- <div class="row my-1">

<div class="col-6">

<label for="ProductImage" style="font-weight: 600;font-size: 14px;color:lightgrey">Product

Image</label>

</div>

<div class="col-6">

<input type="file" #fileSelected pInputText formControlName="productImage" (change)="changeImage($event)"/>

</div>

</div>

<div class="image-preview" style="display: flex;justify-content: center;align-items: center;"\*ngIf="imagePreview !== '' && imagePreview">

<img [src]="imagePreview" [alt]="productForm.value.productName" />

</div> -->

</form>

</div>

<div class="card-footer" style="display:flex;justify-content: center;align-items: center;">

<button type="button" class="btn btn-light" style="border:1px solid black;margin:1%; width:10rem;" (click)="onCancel($event)" >Cancel</button>

<button type="button" class="btn btn-primary" style="border:1px solid black;margin:1%; width:10rem;" (click)="onSave($event)" >Save</button>

</div>

</div>

</div>

</div>

<app-acess-alert [dialog]="dialog"></app-acess-alert>

**ADD-PRODUCT.COMPONENT.CSS**

:host ::ng-deep .p-inputtext{

width:80%!important;

}

:host ::ng-deep .p-dropdown{

width:80%!important;

}

.image-preview{

height:15rem;

margin:1rem;

}

.image-preview img{

height: 100%;

}

**MODELS**

**API-MODELS.TS**

export interface Products{

id:string;

productName:string;

productPrice:string;

category:string;

image:string;

}

export interface Name{

id:string;

productName:string;

}

**AUTH-DATA.MODEL.TS**

export interface AuthData{

email:string;

password:string;

}

**SERVICES**

**AUTH-SERVICE.TS**

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

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

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

import { Subject } from 'rxjs';

import { AuthData } from '../models/auth-data.model';

@Injectable({

providedIn: 'root'

})

export class AuthService {

private isAdmin:boolean =false;

private isAuthenticated: boolean = false;

private adminListener = new Subject<boolean>();

private token: string;

private authStatusListener = new Subject<boolean>();

private tokenTimer: any;

private user:any;

private userListener = new Subject<any>();

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

getToken() {

return this.token;

}

getIsAdmin(){

return this.isAdmin;

}

getIsAuth() {

return this.isAuthenticated;

}

getUserListener(){

return this.userListener.asObservable();

}

getAdminListener(){

return this.adminListener.asObservable();

}

getAuthStatusListener() {

return this.authStatusListener.asObservable();

}

createUser(email: string, password: string) {

const authData: AuthData = { email: email, password: password };

this.http.post("http://localhost:3000/api/user/signUp", authData)

.subscribe(response => {

console.log(response);

})

}

getUserDetails(){

this.http.get("http://localhost:3000/api/user")

.subscribe((res:any)=>{

this.user = res.user;

this.userListener.next(this.user);

})

}

deleteUser(userId:string){

this.http

.delete('http://localhost:3000/api/user/' + userId)

.subscribe(()=>{

const updatedUser = this.user.filter((user:any) => user.\_id !== userId);

this.user.user = updatedUser;

this.userListener.next(this.user);

});

}

loginUser(email: string, password: string) {

const authData: AuthData = { email: email, password: password };

this.http.post<{ token: string, expiresIn: number, userType:string }>("http://localhost:3000/api/user/login", authData)

.subscribe(response => {

const token = response.token;

this.token = token;

if (token) {

const expiresIn = response.expiresIn;

const userType = response.userType;

this.setAuthTimer(expiresIn);

this.isAdmin = response.userType === 'admin'?true:false;

console.log(this.isAdmin);

this.isAuthenticated = true;

this.adminListener.next(this.isAdmin);

this.authStatusListener.next(true);

const now = new Date();

const expirationDate = new Date(now.getTime() + expiresIn \* 1000);

this.saveAuthData(this.token, expirationDate, userType);

this.router.navigate(['home']);

}

});

}

logout() {

this.token = '';

this.isAuthenticated = false;

this.authStatusListener.next(false);

clearTimeout(this.tokenTimer);

this.clearAuthData();

this.router.navigate(['login']);

}

autoAuthUser(){

const authInformation = this.getAuthData();

if(!authInformation){

return;

}

const now = new Date();

const expiresIn = authInformation!.expirationDate.getTime() - now.getTime();

if(expiresIn > 0){

this.token = authInformation!.token;

this.isAuthenticated = true;

this.setAuthTimer(expiresIn/1000);

this.authStatusListener.next(true);

}

}

private setAuthTimer(duration:number){

this.tokenTimer = setTimeout(() => {

this.logout();

}, duration \* 1000);

}

private saveAuthData(token:string, expirationDate:Date, userType:string){

localStorage.setItem("userType", userType);

localStorage.setItem("token", token);

localStorage.setItem("expiration", expirationDate.toISOString());

}

private clearAuthData(){

localStorage.removeItem("userType");

localStorage.removeItem("token");

localStorage.removeItem("expiration");

}

private getAuthData(){

const token = localStorage.getItem("token");

const expirationDate = localStorage.getItem("expiration");

const userType = localStorage.getItem("userType");

if(!token || !expirationDate || !userType){

return;

}

return {

token:token,

expirationDate:new Date(expirationDate),

userType: userType

}

}

}

**AUTH-INTERCEPTOR.TS**

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

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

import { AuthService } from "./auth.service";

@Injectable()

export class AuthInterceptor implements HttpInterceptor {

constructor(private authService: AuthService) { }

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

const authToken = this.authService.getToken();

const authRequest = req.clone({

headers: req.headers.set('Authorization', "Bearer " + authToken)

});

return next.handle(authRequest);

}

}

**AUTH-GUARD.TS**

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

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

import { Observable } from "rxjs";

import { AuthService } from "./auth.service";

@Injectable()

export class AuthGuard implements CanActivate{

constructor(private authService:AuthService, private router:Router){}

canActivate(

route: ActivatedRouteSnapshot,

state: RouterStateSnapshot

): boolean | UrlTree | Observable<boolean | UrlTree> | Promise<boolean | UrlTree> {

debugger

const isAuth = this.authService.getIsAuth();

if(!isAuth){

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

}

return isAuth;

}

}

**API-SERVICE.TS**

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

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

import { Subject } from 'rxjs';

import { Products } from '../models/api-models';

import {map} from 'rxjs/operators';

@Injectable({

providedIn: 'root'

})

export class ApiService {

private productOrder:any;

private productOrderUpdated = new Subject<any>();

private products:any[] =[];

private productUpdated = new Subject<Products[]>();

constructor(private http:HttpClient) { }

getProducts(){

this.http

.get<{message:string,product:any}>('http://localhost:3000/api/products')

.pipe(map((productData)=>{

return productData.product.map((product:any) =>{

return {

productName:product.productName,

category:product.category,

price:product.productPrice,

id:product.\_id,

image:product.image

};

});

}))

.subscribe((transformedData:any)=>{

this.products = transformedData;

this.productUpdated.next([...this.products]);

});

}

getPostUpdateListener(){

return this.productUpdated.asObservable();

}

getProductOrderUpdatedListener(){

return this.productOrderUpdated.asObservable();

}

addProducts(name:string, category:string, price:string){

const product:Products = {id:'', productName:name, productPrice:price, category:category, image:''};

this.http

.post<{message:string, productId:string}>('http://localhost:3000/api/products',product)

.subscribe((data:any)=>{

const productId = data.productId;

product.id= productId;

this.products.push(product);

this.productUpdated.next([...this.products]);

});

}

updateProducts(id:string, name:string, category:string, price:string ){

const product:Products ={ id:id , productName:name, category:category, productPrice:price, image:'' };

this.http

.put<{message:string, productId:string}>('http://localhost:3000/api/products/' + id,product)

.subscribe((data:any)=>{

const updatedProduct = [...this.products];

const oldProductIndex = updatedProduct.findIndex(p => p.id === product.id);

updatedProduct[oldProductIndex] = product;

this.products = updatedProduct;

this.productUpdated.next([...this.products]);

});

}

deleteProducts(productId:string){

this.http

.delete('http://localhost:3000/api/products/' + productId)

.subscribe(()=>{

const updatedProduct = this.products.filter(product => product.id !== productId);

this.products = updatedProduct;

this.productUpdated.next([...this.products]);

});

}

createOrder(item:any){

const order:any = item;

this.http

.post<{message:string, orderId:string}>('http://localhost:3000/api/orders', order)

.subscribe((data:any)=>{

this.productOrder = data;

this.productOrderUpdated.next(this.productOrder);

})

}

getOrder(){

this.http.get("http://localhost:3000/api/orders")

.subscribe((res:any)=>{

this.productOrder = res;

this.productOrderUpdated.next(this.productOrder);

})

}

}

**LOGIN-API.SERVICE.TS**

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

@Injectable({

providedIn: 'root'

})

export class LoginApiService {

constructor() { }

loggedIn:boolean = false;

product:any;

usercart:any;

productPage:any;

allPages:any;

selectedSection:any='Home';

carouselDetails:any = [{

image: "../../../assets/images/Biscuit.jpg",

captionHeader: "Big Biscuit Deals",

captionDescription: "Buy 2 and Get 2 Free..."

},

{

image: "../../../assets/images/Fruits.jpg",

captionHeader: "Fruits at Low Prices",

captionDescription: "Get upto 50% Off on Fruits..."

},

{

image: "../../../assets/images/Vegetables.jpg",

captionHeader: "Vegetables Offer",

captionDescription: "Get upto 40% Off on All Vegetables"

}]

productImages:[

{

id:1,

image:"../../../assets/images/TomatoT.jpg",

productName:'Tomato'

},

{

id:2,

image:"../../../assets/images/PotatoP.jpg",

productName:'Potato'

},

{

id:3,

image:"../../../assets/images/CarrotC.jpg",

productName:'Carrot'

},

{

id:4,

image:"../../../assets/images/OnionO.jpg",

productName:'Onion'

},

{

id:5,

image:"../../../assets/images/AppleA.jpg",

productName:'Apple'

},

{

id:6,

image:"../../../assets/images/OrangeO.jpg",

productName:'Orange'

},

{

id:7,

image:"../../../assets/images/GrapesG.jpg",

productName:'Grapes'

},

{

id:8,

image:"../../../assets/images/MangoM.jpg",

productName:'Mango'

},

{

id:9,

image:"../../../assets/images/CarrotCakeCC.jpg",

productName:'Carrot Cake'

},

{

id:10,

image:"../../../assets/images/ChocolateCakeCC.jpg",

productName:'Chocolate Cake'

},

{

id:11,

image:"../../../assets/images/StrawberryIceCreamSIC.jpg",

productName:'Strawberry IceCream'

},

]

productDetails:any=[

{

id: 1,

productName: 'Tomato',

productPrice: 22,

quantity: 0,

category: 'Vegetables',

image:"../../../assets/images/Tomato.jpg",

isWishlist:false,

},

{

id: 2,

productName: 'Potato',

productPrice: 22,

quantity: 0,

category: 'Vegetables',

image:"../../../assets/images/Tomato.jpg",

isWishlist:false,

},

{

id: 3,

productName: 'Beans',

productPrice: 22,

quantity: 0,

category: 'Vegetables',

image:"../../../assets/images/Tomato.jpg",

isWishlist:false,

},

{

id: 4,

productName: 'Carrot',

productPrice: 22,

quantity: 0,

category: 'Vegetables',

image:"../../../assets/images/Tomato.jpg",

isWishlist:false,

},

]

cartProduct:any=[];

wishProduct:any=[];

}

**APP.MODULE.TS**

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

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

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

import {HttpClientModule, HTTP\_INTERCEPTORS} from '@angular/common/http';

import {InputNumberModule} from 'primeng/inputnumber';

import {InputTextModule} from 'primeng/inputtext';

import {DropdownModule} from 'primeng/dropdown';

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

import {TableModule} from 'primeng/table';

import {DialogModule} from 'primeng/dialog';

import {ProgressSpinnerModule} from 'primeng/progressspinner';

import {InputTextareaModule} from 'primeng/inputtextarea';

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

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

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

import { HomeGroceryComponent } from './components/home-grocery/home-grocery.component';

import { GroceryHeaderComponent } from './components/grocery-header/grocery-header.component';

import { ProductComponent } from './components/product/product.component';

import { ProductSectionComponent } from './components/product-section/product-section.component';

import { AddProductComponent } from './components/add-product/add-product.component';

import { ProductViewComponent } from './components/product-view/product-view.component';

import { EditProductComponent } from './components/edit-product/edit-product.component';

import { AuthInterceptor } from './services/auth-interceptor';

import { CartViewComponent } from './components/cart-view/cart-view.component';

import { OrderViewComponent } from './components/order-view/order-view.component';

import { CarouselSectionComponent } from './components/carousel-section/carousel-section.component';

import { UserViewComponent } from './components/user-view/user-view.component';

import { WishlistItemsComponent } from './components/wishlist-items/wishlist-items.component';

import { AcessAlertComponent } from './components/acess-alert/acess-alert.component';

@NgModule({

declarations: [

AppComponent,

LoginComponent,

HomeGroceryComponent,

GroceryHeaderComponent,

ProductComponent,

ProductSectionComponent,

AddProductComponent,

ProductViewComponent,

EditProductComponent,

CartViewComponent,

OrderViewComponent,

CarouselSectionComponent,

UserViewComponent,

WishlistItemsComponent,

AcessAlertComponent

],

imports: [

BrowserModule,

BrowserAnimationsModule,

AppRoutingModule,

FormsModule,

ReactiveFormsModule,

HttpClientModule,

InputNumberModule,

InputTextModule,

DropdownModule,

TableModule,

DialogModule,

ProgressSpinnerModule,

InputTextareaModule

],

providers: [

{provide:HTTP\_INTERCEPTORS, useClass:AuthInterceptor, multi:true}

],

bootstrap: [AppComponent]

})

export class AppModule { }

**APP-ROUTING-MODULE.TS**

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

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

import { AddProductComponent } from './components/add-product/add-product.component';

import { CartViewComponent } from './components/cart-view/cart-view.component';

import { HomeGroceryComponent } from './components/home-grocery/home-grocery.component';

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

import { OrderViewComponent } from './components/order-view/order-view.component';

import { ProductViewComponent } from './components/product-view/product-view.component';

import { UserViewComponent } from './components/user-view/user-view.component';

import { WishlistItemsComponent } from './components/wishlist-items/wishlist-items.component';

import { AuthGuard } from './services/auth-guard';

const routes: Routes = [

{

path: '',

redirectTo: '/login',

pathMatch: 'full'

},

{

path: 'home',

component: HomeGroceryComponent,

canActivate:[AuthGuard]

},

{

path: 'add',

component: AddProductComponent,

canActivate:[AuthGuard]

},

{

path: 'view',

component: ProductViewComponent,

canActivate:[AuthGuard]

},

{

path: 'cart',

component: CartViewComponent,

canActivate:[AuthGuard]

},

{

path: 'order',

component: OrderViewComponent,

canActivate:[AuthGuard]

},

{

path: 'user-view',

component: UserViewComponent,

canActivate:[AuthGuard]

},

{

path: 'wishlist',

component: WishlistItemsComponent,

canActivate:[AuthGuard]

},

{

path: 'login',

component: LoginComponent

}

];

@NgModule({

imports: [RouterModule.forRoot(routes)],

exports: [RouterModule],

providers:[AuthGuard]

})

export class AppRoutingModule { }

**BACKEND**

**SERVER.JS**

const http = require('http');

const debug = require('debug')("application");

const app = require('./app');

const normalizePort = val =>{

var port = parseInt(val,10);

if(isNaN(port)){

return val;

}

if(port >= 0){

return port;

}

return false;

};

const onError = error=>{

if(error.syscall !== "listen"){

throw error;

}

const bind = typeof addr === "string" ? "pipe " + addr : "port " + port;

switch (error.code){

case "EACCES":

console.error(bind + "requires elevated privileges");

process.exit(1);

break;

case "EADDRINUSE":

console.error(bind + " is already in use");

process.exit(1);

break;

default:

throw error;

}

};

const onListening = () =>{

const addr = server.address();

const bind = typeof addr === "string" ? "pipe " + addr : "port " + port;

debug("Listening on " + bind);

}

const port = process.env.PORT || 3000;

app.set('port', port);

const server = http.createServer(app);

server.on("error", onError);

server.on("Listening", onListening);

server.listen(port);

**APP.JS**

const express = require('express');

const bodyParser = require('body-parser');

const mongoose = require('mongoose');

const PageDetail = require('./models/pageDetail');

const productRoutes = require('./routes/product');

const userRoutes = require('./routes/user');

const orderRoutes = require('./routes/order');

const app = express();

mongoose.connect("mongodb://host.docker.internal:27017/grocery")

.then(() => {

console.log('Connected to the database');

})

.catch(() => {

console.log("Connection failed");

})

app.use(bodyParser.json());

app.use(bodyParser.urlencoded({ extended: false }));

app.use((req, res, next) => {

res.setHeader("Access-Control-Allow-Origin", "\*");

res.setHeader(

"Access-Control-Allow-Headers",

"Origin, X-Requested-With, Content-Type, Accept, Authorization"

);

res.setHeader(

"Access-Control-Allow-Methods",

"GET, POST, PATCH, DELETE,PUT, OPTIONS"

);

next();

});

app.post("/api/pageDetail", (req, res, next) => {

const page = new PageDetail({

pageName: req.body.pageName,

carouselDetails:req.body.carouselDetails

});

page.save().then(page =>{

res.status(201).json({

message: "PageDetail added successfully",

pageDetailId:page.\_id

});

});

})

app.use("/api/products", productRoutes)

app.use("/api/user", userRoutes);

app.use("/api/orders", orderRoutes);

module.exports = app;

**ROUTES/ORDER.JS**

const express = require('express');

const Order = require('../models/order');

const router = express.Router();

router.post("", (req, res, next) => {

const order = new Order({

product:req.body

});

order.save().then(order =>{

res.status(201).json({

message: "Ordered successfully",

orderId:order.\_id

});

});

});

router.get("", (req, res, next) => {

Order.find().then(order =>{

res.status(200).json({

message: 'Order details fetched successfully!',

order: order

});

});

});

module.exports = router;

**ROUTES/PRODUCT.JS**

const express = require('express');

const Product = require('../models/product');

// const verifyAuth = require('../middleware/verify-auth');

const router = express.Router();

const MIME\_TYPE\_MAP = {

'image/png':'png',

'image/jpeg':'jpg',

'image/jpg':'jpg'

};

router.post("", (req, res, next) => {

const product = new Product({

productName: req.body.productName,

category: req.body.category,

productPrice:req.body.productPrice,

userType:req.body.userType

});

if(userType == 'customer'){

res.status(401).json({

message:"Not authorised"

});

}else

{

product.save().then(createdProduct =>{

res.status(201).json({

message: "Product added successfully",

productId:createdProduct.\_id

});

});

}

})

router.put("/:id",(req, res, next) => {

const product = new Product({

\_id:req.body.id,

productName: req.body.productName,

category: req.body.category,

productPrice:req.body.productPrice

});

Product.updateOne({\_id:req.params.id}, product).then(result =>{

res.status(200).json({

message: 'Product updated successfully'

});

});

})

router.get("", (req, res, next) => {

Product.find().then(documents =>{

res.status(200).json({

message: 'Product fetched successfully!',

product: documents

});

});

});

router.delete("/:id",(req, res, next) => {

Product.deleteOne({\_id:req.params.id}).then(result =>{

console.log(result);

res.status(200).json({

message: 'Product deleted successfully'

});

});

});

module.exports = router;

**ROUTES/USER.JS**

const express = require('express');

const bcrypt = require('bcrypt');

const jwt = require('jsonwebtoken');

const User = require('../models/user');

const router = express.Router();

router.post("/signUp", (req, res, next)=>{

bcrypt.hash(req.body.password, 10)

.then(hash=>{

const user = new User({

email:req.body.email,

password:hash,

userType:'customer'

});

user.save()

.then(result =>{

res.status(201).json({

message:"User created successfully!",

result:result

});

});

})

.catch(err=>{

res.status(500).json({

error:err

});

});

});

router.post("/login", (req, res, next)=>{

let fetchedUser;

User.findOne({email:req.body.email})

.then(user=>{

if(!user){

return res.status(404).json({

message:"User does not exit!!"

})

}

fetchedUser = user;

console.log(fetchedUser);

return bcrypt.compare(req.body.password, user.password);

})

.then(result=>{

if(!result){

return res.status(401).json({

message:"Not successful!!"

});

}

const token = jwt.sign(

{email:fetchedUser.email, userId:fetchedUser.\_id},

"sfdhsdjakfdfdskjfahfjdshjkfshf",

{expiresIn:"1h"}

);

res.status(200).json({

token:token,

userType:fetchedUser.userType,

expiresIn:3600

});

})

.catch(err=>{

return res.status(401).json({

message:"Not successful!!"

});

});

})

router.get("", (req, res, next) => {

User.find().then(user =>{

res.status(200).json({

message: 'User details successfully!',

user: user

});

});

});

router.delete("/:id",(req, res, next) => {

User.deleteOne({\_id:req.params.id}).then(result =>{

res.status(200).json({

message: 'User deleted successfully'

});

});

});

module.exports = router;

**MODELS/ORDER.JS**

const mongoose = require('mongoose');

const orderSchema = mongoose.Schema({

product:[{productId:String, productName:String,price:String, quantity:String, category:String }]

});

module.exports = mongoose.model('Order', orderSchema);

**MODELS/PAGEDETAIL.JS**

const mongoose = require('mongoose');

const pageDetailSchema = mongoose.Schema({

pageName:{type:String, required:true },

carouselDetails:{image:String, captionHeader:String,captionDescription:String }

});

module.exports=mongoose.model('PageDetail', pageDetailSchema);

**MODELS/PRODUCT.JS**

**const mongoose = require('mongoose');**

**const productSchema = mongoose.Schema({**

**productName:{type:String, required:true },**

**category:{type:String, required:true },**

**productPrice:{type:String, required:true },**

**image:{type:String}**

**});**

**module.exports=mongoose.model('Product', productSchema);**

**MODELS/USER.JS**

**const mongoose = require('mongoose');**

**const uniqueValidator = require('mongoose-unique-validator');**

**const userSchema = mongoose.Schema({**

**email:{type:String, required:true, unique:true},**

**password:{type:String, required:true},**

**userType:{type:String}**

**});**

**userSchema.plugin(uniqueValidator);**

**module.exports=mongoose.model('User', userSchema);**

**MIDDLEWARE/VERIFY-AUTH.JS**

const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {

try {

const token = req.headers.authorization.split("")[1];

const decodedToken = jwt.verify(token, "sfdhsdjakfdfdskjfahfjdshjkfshf");

req.userData = {email:decodedToken.email, id:decodedToken.userId, userType:decodedToken.userType};

next();

} catch (err){

res.status(401).json({message:"Not Authorized!!!"});

}

}