**Frontend-Angular**

* **Admin:**

1. **Component: -**

**Navbar.component.html**

<navclass=”navbar navbar-expand-lg navbar-light bg-light”>

    <buttonclass=”navbar-toggler”type=”button”data-toggle=”collapse”data-target=”#navbarSupportedContent”

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

        <spanclass=”navbar-toggler-icon”></span>

    </button>

    <divclass=”collapse navbar-collapse”id=”navbarSupportedContent”>

        <aclass=”navbar-brand”routerLink=”/”><imgsrc=”assets/Images/Medicare-Logo.png”alt=”Logo”

                style=”width:100px;height:45px;”></a>

        <ulclass=”navbar-nav mr-auto”>

            <liclass=”nav-item active”>

                <aclass=”nav-link”routerLink=”/get/all/class/All-Medicines”>Medicines<span

                        class=”sr-only”>(current)</span></a>

            </li>

            <liclass=”nav-item”>

                <aclass=”nav-link”href=”#”>Lab Tests</a>

            </li>

            <liclass=”nav-item dropdown”>

                <aclass=”nav-link dropdown-toggle”href=”#”id=”navbarDropdown”role=”button”data-toggle=”dropdown”

                    aria-haspopup=”true”aria-expanded=”false”>

                    Healthcare

                </a>

                <divclass=”dropdown-menu”aria-labelledby=”navbarDropdown”>

                    <aclass=”dropdown-item”href=”#”>Covid Essentials</a>

                    <aclass=”dropdown-item”href=”#”>Personal Care</a>

                    <divclass=”dropdown-divider”></div>

                    <aclass=”dropdown-item”href=”#”>Surgical Care</a>

                </div>

            </li>

            <liclass=”nav-item”>

                <aclass=”nav-link”href=”#”>Health Blogs</a>

            </li>

        </ul>

        <ulclass=”navbar-nav ml-auto”>

            <liclass=”nav-item mr-1”>

                <arouterLink=”/get/cart/details”role=”button”class=”btnbtn-info”><small>Cart total: {{totalPrice |

                        currency:

                        ‘INR’}} </small><Iclass=’fas fa-cart-arrow-down’></i>&nbsp;<span

                        class=”badge badge-danger”>{{totalQuantity}}</span></a>

            </li>

            <liclass=”nav-item mr-1”>

                <a\*ngIf=”!loginService.isLoggedIn()”routerLink=”/user/signup”role=”button”class=”btnbtn-link”>Sign

                    Up</a>

            </li>

            <liclass=”nav-item”>

                <a\*ngIf=”!loginService.isLoggedIn()”routerLink=”/user/login”role=”button”

                    class=”btnbtn-outline-success”>Login</a>

            </li>

            <liclass=”nav-item mr-1”>

                <aclass=”btnbtn-link”\*ngIf=”loginService.isLoggedIn()”role=”button”(click)=”home()”><b>Home</b></a>

            </li>

            <liclass=”nav-item”>

                <aclass=”btnbtn-info”\*ngIf=”loginService.isLoggedIn()”role=”button”(click)=”logout()”>Logout</a>

            </li>

        </ul>

    </div>

</nav>

**Navbar.component.ts**

import{ ComponentFixture, TestBed } from‘@angular/core/testing’;

import{ NavbarComponent } from‘./navbar.component’;

describe(‘NavbarComponent’, () => {

  letcomponent:NavbarComponent;

  letfixture:ComponentFixture<NavbarComponent>;

  beforeEach(async () => {

    awaitTestBed.configureTestingModule({

      declarations: [ NavbarComponent ]

    })

    .compileComponents();

    fixture=TestBed.createComponent(NavbarComponent);

    component=fixture.componentInstance;

    fixture.detectChanges();

  });

  it(‘should create’, () => {

    expect(component).toBeTruthy();

  });

});

**Admin-dashboard.component.html**

<br>

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col”>

            <divclass=”cardbg-light”>

                <divclass=”card-header”>

                    <h3class=”text-justify”>Welcome Admin: <spanclass=”text-primary”>{{name}}</span>

                        &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<spanclass=”offset-4”>{{username}}</span></h3>

                </div>

                <divclass=”card-body”>

                    <divclass=”card-deck”>

                        <divclass=”cardbg-success”>

                            <imgclass=”card-img-top”src=”assets/Images/add-medicine.png”alt=”add-medicine.png”

                                style=”width:100%; height:250px;”>

                            <divclass=”card-body text-center”>

                                <h4class=”card-title”>Add Medicine</h4>

                                <pclass=”card-text”>Add a new medicine to our store.</p>

                            </div>

                            <divclass=”card-footer”>

                                <arouterLink=”/admin/add-medicine”role=”button”class=”btnbtn-warning btn-block”>Add

                                    Medicine</a>

                            </div>

                        </div>

                        <divclass=”cardbg-info”>

                            <imgclass=”card-img-top”src=”assets/Images/all-medicines.png”alt=”all-medicines.png”

                                style=”width:100%; height:250px;”>

                            <divclass=”card-body text-center”>

                                <h4class=”card-title”>All Medicines</h4>

                                <pclass=”card-text”>Display all the medicine’s in our store.</p>

                            </div>

                            <divclass=”card-footer”>

                                <arouterLink=”/admin/get/all/medicines”role=”button”

                                    class=”btnbtn-danger btn-block”>Show All Medicines</a>

                            </div>

                        </div>

                        <divclass=”cardbg-secondary”>

                            <imgclass=”card-img-top”src=”assets/Images/order.png”alt=”order.png”

                                style=”width:100%; height:250px;”>

                            <divclass=”card-body text-center”>

                                <h4class=”card-title”>All Orders</h4>

                                <pclass=”card-text”>Display all the order’s created by customers.</p>

                            </div>

                            <divclass=”card-footer”>

                                <arouterLink=”/admin/all/user-orders”role=”button”class=”btnbtn-dark btn-block”>Show

                                    All Orders</a>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

**admin-dashboard.component.ts**

import{ Component } from‘@angular/core’;

import{ LoginService } from‘src/app/Services/login.service’;

@Component({

  selector:‘app-admin-dashboard’,

  templateUrl:‘./admin-dashboard.component.html’,

  styleUrls: [‘./admin-dashboard.component.css’]

})

exportclassAdminDashboardComponent {

  username!:string;

  name!:string;

  constructor(privateloginService:LoginService) {

    this.username=this.loginService.getUserDetails().username;

    this.name=this.loginService.getUserDetails().firstName+‘‘+this.loginService.getUserDetails().lastName;

  }

}

**Admin-login.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-lg-4 col-md-8 mx-auto”>

            <divclass=”card mt-5 bg-warning”>

                <divclass=”card-body”>

                    <divclass=”text-center”>

                        <h4class=”card-title”>Admin Login</h4>

                        <p>Sign in to continue.</p>

                    </div>

                    <hr>

                    <form#login=”ngForm”(ngSubmit)=”onSubmit()”>

                        <divclass=”form-group”>

                            <labelfor=”email”><b>Username:</b></label><inputtype=”text”name=”username”

                                #username=”ngModel”class=”form-control”placeholder=”Enter Username”id=”email”

                                [(ngModel)]=”credentials.username”requiredngModel>

                            <span\*ngIf=”username.invalid&&username.touched”style=”color:red”>Please enter a valid

                                username.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”pwd”><b>Password:</b></label><inputtype=”password”name=”password”

                                #password=”ngModel”minlength=”6”class=”form-control”placeholder=”Enter password”

                                id=”pwd”[(ngModel)]=”credentials.password”requiredngModel>

                            <span\*ngIf=”password.invalid&&password.touched”style=”color: red”>Please enter a valid

                                password.</span>

                        </div>

                        <buttontype=”submit”class=”btnbtn-success btn-block”

                            [disabled]=”login.invalid”>Login</button>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**Admin-Login.component.ts**

import{ Component } from‘@angular/core’;

import{ Router } from‘@angular/router’;

import{ Credentials } from‘src/app/credentials’;

import{ LoginService } from‘src/app/Services/login.service’;

@Component({

  selector:‘app-admin-login’,

  templateUrl:‘./admin-login.component.html’,

  styleUrls: [‘./admin-login.component.css’]

})

exportclassAdminLoginComponent {

  credentials:Credentials=newCredentials();

  constructor(privateloginService:LoginService, privaterouter:Router) { }

  onSubmit() {

    this.loginService.generateToken(this.credentials).subscribe({

      next: (response) => {

        //user is logged in

        this.loginService.loginUser(response.token);

        this.loginService.getCurrentUser().subscribe({

          next: (admin) => {

            this.loginService.setUserDetails(admin);

            //redirect: to admin dashboard

            if (this.loginService.getUserRole() ==‘ADMIN’) {

              this.router.navigate([‘/admin-dashboard’]);

            }

          }, error: (error) => {

            console.log(error);

          }

        })

      }, error: (error) => {

        console.log(error);

        alert(‘Invalid Credentials!’);

      }

    })

  }

}

[**Home.component.html**](http://Home.component.html)

<br>

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-6 mx-auto”>

            <h4>What are you looking for?</h4><br>

            <divclass=”input-group mb-3”>

                <divclass=”input-group-prepend”>

                    <spanclass=”input-group-text”><imgsrc=”assets/Images/search-icon.png”alt=””></span>

                </div>

                <inputtype=”text”class=”form-control”placeholder=”Search medicine, health products and more...”

                    [(ngModel)]=”name”(keydown.enter)=”onSearch(name)”>

            </div>

        </div>

    </div>

</div>

<divid=”img”class=”carousel slide”data-ride=”carousel”>

    <ulclass=”carousel-indicators”>

        <lidata-target=”#img”data-slide-to=”0”class=”active”></li>

        <lidata-target=”#img”data-slide-to=”1”></li>

        <lidata-target=”#img”data-slide-to=”2”></li>

        <lidata-target=”#img”data-slide-to=”3”></li>

    </ul>

    <divclass=”carousel-inner”>

        <divclass=”carousel-item active”>

            <imgsrc=”assets/Images/pic1.jpg”alt=”pic1.jpg”>

        </div>

        <divclass=”carousel-item”>

            <imgsrc=”assets/Images/pic2.jpg”alt=”pic2.jpg”>

        </div>

        <divclass=”carousel-item”>

            <imgsrc=”assets/Images/pic3.jpg”alt=”pic3.jpg”>

        </div>

        <divclass=”carousel-item”>

            <imgsrc=”assets/Images/pic4.jpg”alt=”pic4.jpg”>

        </div>

    </div>

    <aclass=”carousel-control-prev”href=”#img”data-slide=”prev”>

        <spanclass=”carousel-control-prev-icon”></span>

    </a>

    <aclass=”carousel-control-next”href=”#img”data-slide=”next”>

        <spanclass=”carousel-control-next-icon”></span>

    </a>

</div><br>

<divclass=”container-fluid”>

    <divclass=”row”>

        <divclass=”col”>

            <h3class=”ml-4 pl-5”>Browse medicines & health products</h3>

            <br>

            <h6class=”ml-4 pl-5”>By health concerns</h6>

        </div>

    </div>

</div>

<divclass=”container-fluid “>

    <divclass=”row”>

        <divclass=”col mb-5”>

            <divid=”product”class=”carousel slide”data-ride=”carousel”data-interval=”false”>

                <divclass=”carousel-inner”>

                    <divclass=”carousel-item active”>

                        <divclass=”card-deck col-11 mx-auto”>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/cardiac.png”alt=”cardiac.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[0]}}”>Cardiac Care</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/diabetic.png”alt=”diabetic.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[1]}}”>Diabetic Care</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/gastro.png”alt=”gastro.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[2]}}”>Gastro & Liver Care</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/urology.png”alt=”urology.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[3]}}”>Urology Care</a>

                                </div>

                            </div>

                        </div>

                    </div>

                    <divclass=”carousel-item”>

                        <divclass=”card-deck col-11 mx-auto”>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/ai.png”alt=”ai.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[4]}}”>Anti Infectives</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/gynaecology.png”alt=”gynaecology.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[5]}}”>Gynaecology Care</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/ortho.png”alt=”ortho.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[6]}}”>Pain Relief</a>

                                </div>

                            </div>

                            <divclass=”cardbg-light”>

                                <imgclass=”card-img-top”src=”assets/Images/vitamins.png”alt=”vitamins.png”

                                    style=”width:100%; height:130px;”>

                                <divclass=”card-body text-center mt-1”>

                                    <arouterLink=”/show/product/class/{{medicine[7]}}”>Daily Health Care</a>

                                </div>

                            </div>

                        </div>

                    </div>

                    <aclass=”carousel-control-prev”href=”#product”data-slide=”prev”>

                        <spanclass=”carousel-control-prev-icon”></span>

                    </a>

                    <aclass=”carousel-control-next”href=”#product”data-slide=”next”>

                        <spanclass=”carousel-control-next-icon”></span>

                    </a>

                </div>

            </div>

        </div>

    </div>

</div>

[**home.component.ts**](http://home.component.ts)

import{ ComponentFixture, TestBed } from‘@angular/core/testing’;

import{ HomeComponent } from‘./home.component’;

describe(‘HomeComponent’, () => {

  letcomponent:HomeComponent;

  letfixture:ComponentFixture<HomeComponent>;

  beforeEach(async () => {

    awaitTestBed.configureTestingModule({

      declarations: [ HomeComponent ]

    })

    .compileComponents();

    fixture=TestBed.createComponent(HomeComponent);

    component=fixture.componentInstance;

    fixture.detectChanges();

  });

  it(‘should create’, () => {

    expect(component).toBeTruthy();

  });

});

y

publicinterfaceAdminCredRepoextendsMongoRepository<AdminCred, String> {

    publicAdminCredfindByUsername(Stringusername);

}

**Add-product.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-6 mx-auto mt-3 mb-4”>

            <divclass=”card”>

                <divclass=”card-header”>

                    <h4class=”text-center”>Add a Medicine</h4>

                </div>

                <divclass=”card-body”>

                    <form#registerationForm=”ngForm”>

                        <label>Medicine Continuity:</label>&nbsp;&nbsp;&nbsp;

                        <divclass=”form-check-inline”>

                            <labelclass=”form-check-label”>

                                <inputtype=”radio”class=”form-check-input”name=”optradio”value=”true”

                                    [(ngModel)]=”product.available”>Continued

                            </label>

                        </div>

                        <divclass=”form-check-inline”>

                            <labelclass=”form-check-label”>

                                <inputtype=”radio”class=”form-check-input”name=”optradio”value=”false”

                                    [(ngModel)]=”product.available”>Discontinued

                            </label>

                        </div>

                        <hr>

                        <divclass=”form-group”>

                            <labelfor=”mname”>Medicine Name:</label>

                            <inputtype=”text”class=”form-control”required#mname=”ngModel”

                                placeholder=”Enter medicine name”id=”mname”name=”name”[(ngModel)]=”product.name”

                                ngModel>

                            <span\*ngIf=”mname.invalid&&mname.touched”style=”color: red”>Please enter a valid

                                medicine name.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”bname”>Medicine Brand Name:</label>

                            <inputtype=”text”required#bname=”ngModel”class=”form-control”

                                placeholder=”Enter medicine brand name”id=”bname”name=”brand”

                                [(ngModel)]=”product.brand”ngModel>

                            <span\*ngIf=”bname.invalid&&bname.touched”style=”color: red”>Please enter a valid

                                medicine brand .</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”mctg”>Medicine Category:</label>

                            <inputtype=”text”class=”form-control”required#mctg=”ngModel”

                                placeholder=”Enter medicine category”id=”mctg”name=”category”

                                [(ngModel)]=”product.category”ngModel>

                            <span\*ngIf=”mctg.invalid&&mctg.touched”style=”color:red”>Please enter a valid

                                medicine category.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”desc”>Medicine Description:</label>

                            <inputtype=”text”class=”form-control”required#desc=”ngModel”

                                placeholder=”Enter medicine description”id=”desc”name=”description”

                                [(ngModel)]=”product.description”ngModel>

                            <span\*ngIf=”desc.invalid&&desc.touched”style=”color:red”>Please enter a valid

                                medicine description.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”scomp”>Salt Composition:</label>

                            <inputtype=”text”required#scomp=”ngModel”class=”form-control”

                                placeholder=”Enter medicine salt composition”id=”scomp”name=”salt”

                                [(ngModel)]=”product.salt”ngModel>

                            <span\*ngIf=”scomp.invalid&&scomp.touched”style=”color: red”>Please enter a valid

                                salt name.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”stk”>Total Stock (In Strip):</label>

                            <inputtype=”number”required#stk=”ngModel”class=”form-control”

                                placeholder=”Enter total medicine stock”id=”stk”name=”totalAvailable”

                                [(ngModel)]=”product.totalAvailable”ngModel>

                            <span\*ngIf=”stk.invalid&&stk.touched”style=”color: red”>Please enter a valid

                                total stock.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”pc”>Unit Price:</label>

                            <divclass=”input-group”>

                                <divclass=”input-group-prepend”>

                                    <spanclass=”input-group-text”>&#8377;</span>

                                </div>

                                <inputtype=”number”required#pc=”ngModel”class=”form-control”

                                    placeholder=”Enter medicine unit price”id=”pc”name=”price”

                                    [(ngModel)]=”product.price”ngModel>

                            </div>

                            <span\*ngIf=”pc.invalid&&pc.touched”style=”color: red”>Please enter a valid

                                unit price.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”img”>Select Medicine Image:</label>

                            <inputtype=”file”class=”form-control-file border”id=”img”

                                (change)=”onChangeFileField($event)”>

                        </div>

                        <hr>

                        <divclass=”text-center”>

                            <divclass=”btn-group”>

                                <button(click)=”onSubmit()”class=”btnbtn-primary”

                                    [disabled]=”registerationForm.invalid”data-toggle=”modal”

                                    data-target=”#myModal”>Add Medicine</button>

                                <!–The Modal 🡪

                                <divclass=”modal fade”id=”myModal”>

                                    <divclass=”modal-dialog modal-dialog-centered”>

                                        <divclass=”modal-content”>

                                            <!–Modal Header 🡪

                                            <divclass=”modal-header”>

                                                <h4class=”modal-title”>Add a new medicine</h4>

                                                <buttontype=”button”class=”close”(click)=”onClick()”

                                                    data-dismiss=”modal”>&times;</button>

                                            </div>

                                            <!–Modal body 🡪

                                            <divclass=”modal-body”>

                                                <span\*ngIf=”isValid”style=”color:green”>{{message}}</span>

                                                <span\*ngIf=”!isValid”style=”color: red”>{{message}}</span>

                                            </div>

                                            <!–Modal footer 🡪

                                            <divclass=”modal-footer”>

                                                <buttontype=”button”class=”btnbtn-secondary”data-dismiss=”modal”

                                                    (click)=”onClick()”>Close</button>

                                            </div>

                                        </div>

                                    </div>

                                </div>

                                <buttontype=”reset”class=”btnbtn-danger”>Reset</button>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**Add-product.component.ts**

import{ Component } from‘@angular/core’;

import{ Router } from‘@angular/router’;

import{ Product } from‘src/app/product’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-add-product’,

  templateUrl:‘./add-product.component.html’,

  styleUrls: [‘./add-product.component.css’]

})

exportclassAddProductComponent {

  constructor(privateuserService:UserService, privaterouter:Router) { }

  product:Product=newProduct();

  file!:Blob;

  isValid!:ompone;

  message!:string;

  onSubmit() {

    this.userService.addMedicine(this.product, this.file).subscribe({

      next: (response) => {

        this.isValid=true;

        this.message=“Medicine added successfully!”

      }, error: (error) => {

        this.isValid=false;

        this.message=‘Something went wrong!’

      }

    })

  }

  onChangeFileField(event:any) {

    this.file=event.target.files[0];

  }

  onClick() {

    this.router.navigate([‘/admin-dashboard’]);

  }

}

**All-order.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col mt-2”>

            <divclass=”table-responsive”>

                <tableclass=”table table-bordered”>

                    <theadclass=”thead-dark”>

                        <tr>

                            <th>S.No.</th>

                            <th>Order-Date</th>

                            <th>Order-Id</th>

                            <th>Order-Details</th>

                            <th>User-Id</th>

                            <th>Full Name</th>

                            <th>Shipping Address</th>

                            <th>State</th>

                            <th>Contact</th>

                            <th>Paid Amount</th>

                            <th>Payment Mode</th>

                            <th>Order-Status</th>

                            <th>Action</th>

                        </tr>

                    </thead>

                    <tbody>

                        <trclass=”table-secondary”\*ngFor=”let o of orders”>

                            <td></td>

                            <td>{{o.date}}</td>

                            <td>{{o.oid}}</td>

                            <td><br><br><buttonclass=”btnbtn-info”(click)=”getOrderDetails(o.oid)”>Show

                                    Order</button>

                            </td>

                            <td>{{o.username}}</td>

                            <td>{{o.firstName}} {{o.lastName}}</td>

                            <td>{{o.address}} {{o.district}}-{{o.pinCode}}</td>

                            <td>{{o.state}}</td>

                            <td>{{o.contact}}</td>

                            <td>{{o.paidAmount | currency: ‘INR’}}</td>

                            <td>{{o.paymentMode}}</td>

                            <td>{{o.status}}</td>

                            <td><br><br><buttonclass=”btnbtn-danger”(click)=”deleteOrder(o.oid)”>Delete

                                    Order</button></td>

                        </tr>

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

**All-order.component.ts**

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

import{ Router } from‘@angular/router’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-all-orders’,

  templateUrl:‘./all-orders.component.html’,

  styleUrls: [‘./all-orders.component.css’]

})

exportclassAllOrdersComponentimplementsOnInit {

  orders:any[] = [];

  constructor(privateuserService:UserService, privaterouter:Router) { }

  ngOnInit():void {

    this.getAllUserOrders();

  }

  getAllUserOrders() {

    this.userService.getAllOrders().subscribe({

      next: (data) => {

        this.orders=data;

      }, error: (error) => {

        console.log(error);

        alert(‘No Order found’);

      }

    })

  }

  getOrderDetails(oid:number) {

    leturl=‘/order/details/’+oid;

    this.router.navigateByUrl(url);

  }

  deleteOrder(oid:number) {

    this.userService.deleteOrder(oid).subscribe({

      next: (data) => {

        this.getAllUserOrders();

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**Update-product.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-6 mx-auto mt-3 mb-4”>

            <divclass=”card”>

                <divclass=”card-header”>

                    <h4class=”text-center”>Update Medicine</h4>

                </div>

                <divclass=”card-body”>

                    <form#registerationForm=”ngForm”(ngSubmit)=”updateProduct()”>

                        <divclass=”form-group”>

                            <labelfor=”mname”>Medicine Name:</label>

                            <inputtype=”text”class=”form-control”required#mname=”ngModel”

                                placeholder=”Enter medicine name”id=”mname”name=”name”[(ngModel)]=”product.name”

                                ngModel>

                            <span\*ngIf=”mname.invalid&&mname.touched”style=”color: red”>Please enter a valid

                                medicine name.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”bname”>Medicine Brand Name:</label>

                            <inputtype=”text”required#bname=”ngModel”class=”form-control”

                                placeholder=”Enter medicine brand name”id=”bname”name=”brand”

                                [(ngModel)]=”product.brand”ngModel>

                            <span\*ngIf=”bname.invalid&&bname.touched”style=”color: red”>Please enter a valid

                                medicine brand .</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”mctg”>Medicine Category:</label>

                            <inputtype=”text”class=”form-control”required#mctg=”ngModel”

                                placeholder=”Enter medicine category”id=”mctg”name=”category”

                                [(ngModel)]=”product.category”ngModel>

                            <span\*ngIf=”mctg.invalid&&mctg.touched”style=”color:red”>Please enter a valid

                                medicine category.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”desc”>Medicine Description:</label>

                            <inputtype=”text”class=”form-control”required#desc=”ngModel”

                                placeholder=”Enter medicine description”id=”desc”name=”description”

                                [(ngModel)]=”product.description”ngModel>

                            <span\*ngIf=”desc.invalid&&desc.touched”style=”color:red”>Please enter a valid

                                medicine description.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”scomp”>Salt Composition:</label>

                            <inputtype=”text”required#scomp=”ngModel”class=”form-control”

                                placeholder=”Enter medicine salt composition”id=”scomp”name=”salt”

                                [(ngModel)]=”product.salt”ngModel>

                            <span\*ngIf=”scomp.invalid&&scomp.touched”style=”color: red”>Please enter a valid

                                salt name.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”stk”>Total Stock (In Strip):</label>

                            <inputtype=”number”required#stk=”ngModel”class=”form-control”

                                placeholder=”Enter total medicine stock”id=”stk”name=”totalAvailable”

                                [(ngModel)]=”product.totalAvailable”ngModel>

                            <span\*ngIf=”stk.invalid&&stk.touched”style=”color: red”>Please enter a valid

                                total stock.</span>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”pc”>Unit Price:</label>

                            <divclass=”input-group”>

                                <divclass=”input-group-prepend”>

                                    <spanclass=”input-group-text”>&#8377;</span>

                                </div>

                                <inputtype=”number”required#pc=”ngModel”class=”form-control”

                                    placeholder=”Enter medicine unit price”id=”pc”name=”price”

                                    [(ngModel)]=”product.price”ngModel>

                            </div>

                            <span\*ngIf=”pc.invalid&&pc.touched”style=”color: red”>Please enter a valid

                                unit price.</span>

                        </div>

                        <hr>

                        <divclass=”text-center”>

                            <buttontype=”submit”class=”btnbtn-primary”[disabled]=”registerationForm.invalid”

                                data-toggle=”modal”data-target=”#myModal”>Update Medicine</button>

                            <!–The Modal 🡪

                            <divclass=”modal fade”id=”myModal”>

                                <divclass=”modal-dialog modal-dialog-centered”>

                                    <divclass=”modal-content”>

                                        <!–Modal Header 🡪

                                        <divclass=”modal-header”>

                                            <h4class=”modal-title”>Update a medicine</h4>

                                            <buttontype=”button”class=”close”(click)=”onClick()”

                                                data-dismiss=”modal”>&times;</button>

                                        </div>

                                        <!–Modal body 🡪

                                        <divclass=”modal-body”>

                                            <span\*ngIf=”isValid”style=”color:green”>{{message}}</span>

                                            <span\*ngIf=”!isValid”style=”color: red”>{{message}}</span>

                                        </div>

                                        <!–Modal footer 🡪

                                        <divclass=”modal-footer”>

                                            <buttontype=”button”class=”btnbtn-secondary”data-dismiss=”modal”

                                                (click)=”onClick()”>Close</button>

                                        </div>

                                    </div>

                                </div>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**Update-product.component.ts**

import{ Component } from‘@angular/core’;

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

import{ Product } from‘src/app/product’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-update-product’,

  templateUrl:‘./update-product.component.html’,

  styleUrls: [‘./update-product.component.css’]

})

exportclassUpdateProductComponent {

  pid!:number;

  product:Product=newProduct();

  isValid!:ompone;

  message!:string;

  constructor(privateroute:ActivatedRoute, privateuserService:UserService, privaterouter:Router) {

    this.pid=this.route.snapshot.params[‘pid’];

    this.getProduct();

  }

  getProduct() {

    this.userService.findById(this.pid).subscribe({

      next: (data) => {

        this.product=data;

      }, error: (error) => {

        console.log(error);

      }

    })

  }

  updateProduct() {

    this.userService.updateMedicine(this.pid, this.product).subscribe({

      next: (data) => {

        this.isValid=true;

        this.message=‘Medicine details updated successfully!’;

      }, error: (error) => {

        this.isValid=false;

        this.message=‘Something went wrong!’;

      }

    })

  }

  onClick() {

    this.router.navigate([‘/admin/get/all/medicines’]);

  }

}

**Show-all-products.component.html**

<br>

<divclass=”container”>

    <navclass=”navbar navbar-expand-smbg-dark navbar-dark”>

        <aclass=”navbar-brand”routerLink=”/get/all/medicines”(click)=”onClick()”>All Medicines</a>

        <divclass=”dropdowndropright”>

            <buttontype=”button”class=”btnbtn-info dropdown-toggle”data-toggle=”dropdown”>

                Filter by

            </button>

            <divclass=”dropdown-menu”>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Anti Infectives’)”>Anti Infectives</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Anti Hypertensives’)”>Anti Hypertensives</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Anti Diabetic’)”>Anti Diabetics</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Gastro Intestinal’)”>Gastro Intestinal</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Urology’)”>Urology</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Gynaecological’)”>Gynaecological</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Analgesics’)”>Analgesics</a>

                <aclass=”dropdown-item”(click)=”getProductByCategory(‘Vitamins’)”>Vitamins</a>

            </div>

        </div>

        <ulclass=”navbar-nav ml-auto mx-4”>

            <liclass=”nav-item dropdown”>

                <aclass=”nav-link dropdown-toggle”href=”#”id=”navbardrop”data-toggle=”dropdown”>

                    Sort by

                </a>

                <divclass=”dropdown-menu”>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceLowToHigh()”>Price: Low To High</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceHighToLow()”>Price: High To Low</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameAscending()”>Name: A-Z</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameDescending()”>Name: Z-A</a>

                </div>

            </li>

        </ul>

        <formclass=”form-inline”(ngSubmit)=”getProductByName()”>

            <inputclass=”form-control mr-sm-2”type=”text”placeholder=”Search”[(ngModel)]=”medicineName”

                name=”medicineName”>

            <buttonclass=”btnbtn-success”type=”submit”>Search</button>

        </form>

    </nav>

    <divclass=”table-responsive”>

        <tableclass=”table table-light table-bordered table-hover”>

            <theadclass=”thead-light text-center”>

                <tr>

                    <th>Medicine Image</th>

                    <th>Medicine Name</th>

                    <th>Brand</th>

                    <th>Category</th>

                    <th>Salt</th>

                    <th>Unit Price(in &#8377😉</th>

                    <th>Total Stock</th>

                    <th>Action</th>

                    <th>Activate</th>

                </tr>

            </thead>

            <tbody>

                <tr\*ngFor=”let p of product | paginate :

                                                {

                                                    itemsPerPage: tableSize,

                                                    currentPage: page,

                                                    totalItems: count

                                                }”>

                    <td>

                        <imgclass=”img-thumbnail”width=”250”height=”250”[src]=”p.img”>

                    </td>

                    <td>{{p.name}}</td>

                    <td>{{p.brand}}</td>

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

                    <td>{{p.salt}}</td>

                    <td>&#8377; {{p.price}}</td>

                    <td>{{p.totalAvailable}}</td>

                    <td>

                        <divclass=”btn-group”>

                            <buttontype=”button”class=”btnbtn-outline-success”

                                (click)=”updateProduct(p.pid)”>Update</button>

                            <buttontype=”button”class=”btnbtn-outline-danger”

                                (click)=”deleteProduct(p.pid)”>Delete</button>

                        </div>

                    </td>

                    <td>

                        <labelclass=”switch”>

                            <inputtype=”checkbox”name=”available”[(ngModel)]=”p.available”

                                (change)=”onActivate(p.pid,p)”>

                            <spanclass=”slider round”></span>

                        </label>

                    </td>

                </tr>

            </tbody>

        </table>

        <divclass=”d-flex justify-content-center”>

            <pagination-controlspreviousLabel=”Prev”nextLabel=”Next”(pageChange)=”onTableDataChange($event)”>

            </pagination-controls>

        </div>

    </div>

</div>

**Show-all-product.component.ts**

import{ Component } from‘@angular/core’;

import{ Router } from‘@angular/router’;

import{ Product } from‘src/app/product’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-show-all-products’,

  templateUrl:‘./show-all-products.component.html’,

  styleUrls: [‘./show-all-products.component.css’]

})

exportclassShowAllProductsComponent {

  constructor(privateuserService:UserService, privaterouter:Router) {

    this.getAllProducts();

  }

  product!:Product[];

  medicineName!:string;

  prod:Product=newProduct();

  page:number=1;

  count:number=0;

  tableSize:number=7;

  event:any;

  onTableDataChange(event:any) {

    this.page=event;

  }

  getAllProducts() {

    this.userService.getAllMedicine().subscribe({

      next: (data) => {

        this.product=data;

        this.product.forEach((p) => {

          p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

        })

      }, error: (error) => {

        console.log(error);

        alert(‘No Medicines Found’);

      }

    })

  }

  getProductByName() {

    this.onTableDataChange(this.event);

    this.userService.getMedicineByName(this.medicineName).subscribe({

      next: (data) => {

        this.product=data;

        this.product.forEach((p) => {

          p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

        })

      }, error: (error) => {

        console.log(error);

        alert(‘No Medicines Found’);

      }

    })

  }

  sortByPriceLowToHigh() {

    this.product.sort((a, b) =>a.price–b.price);

  }

  sortByPriceHighToLow() {

    this.product.sort((a, b) =>b.price–a.price);

  }

  sortByNameAscending() {

    this.product.sort((a, b) =>a.name.localeCompare(b.name));

  }

  sortByNameDescending() {

    this.product.sort((a, b) =>b.name.localeCompare(a.name));

  }

  getProductByCategory(category:string) {

    this.onTableDataChange(this.event);

    this.userService.getMedicineByCategory(category).subscribe({

      next: (data) => {

        this.product=data;

        this.product.forEach((p) => {

          p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

        })

      }, error: (error) => {

        console.log(error);

        alert(‘No Medicines Found’);

      }

    })

  }

  deleteProduct(pid:number) {

    this.userService.deleteMedicine(pid).subscribe({

      next: (data) => {

        this.getAllProducts();

      }, error: (error) => {

        console.log(error);

      }

    })

  }

  updateProduct(pid:number) {

    leturl=“/admin/update/medicine/”+pid;

    this.router.navigateByUrl(url);

  }

  onClick() {

    window.location.reload();

  }

  onActivate(pid:number, p:Product) {

    this.prod=p;

    this.userService.setAvailable(pid, this.prod).subscribe({

      next: (data) => {

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**Get-product.component.html**

<div>

    <navclass=”navbar navbar-expand-smbg-light justify-content-center”>

        <divclass=”col-6”>

            <inputtype=”text”class=”form-control”placeholder=”Search medicine, health products and more...”

                [(ngModel)]=”name”(keydown.enter)=”onSearch(name)”>

        </div>

    </nav>

</div>

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-sm-12 col-md-10 col-lg-8 mx-auto”>

            <br>

            <spanstyle=”font-size: 1.5rem;”>Products</span>

            <spanclass=”float-right mt-2”><aclass=”btnbtn-light btn-sm dropdown-toggle”href=”#”

                    data-toggle=”dropdown”role=”button”>

                    Sort by

                </a>

                <divclass=”dropdown-menu”>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceLowToHigh()”>Price: Low To High</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceHighToLow()”>Price: High To Low</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameAscending()”>Name: A-Z</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameDescending()”>Name: Z-A</a>

                </div>

            </span>

            <divclass=”card flex-row mt-3 bg-light”\*ngFor=”let p of product | paginate :

                                                                                {

                                                                                    itemsPerPage: tableSize,

                                                                                    currentPage: page,

                                                                                    totalItems: count

                                                                                }”>

                <img[src]=”p.img”class=”card-img-left img-thumbnail”width=”200”height=”200”>

                <divclass=”card-body”>

                    <h4class=”card-title”>{{p.name}} <spanclass=”float-right”>&#8377; {{p.price}}</span></h4>

                    <pclass=”text-muted”>By {{p.brand}}</p>

                    <small>Description: {{p.description}}</small>

                    <br>

                    <small>Contains: <spanclass=”font-weight-bold”>{{p.salt}}</span></small>

                    <buttonclass=”btnbtn-primary float-right mt-2”\*ngIf=”p.available”(click)=”addToCart(p)”>Add to

                        cart</button>

                    <buttonclass=”btnbtn-secondary float-right mt-2”\*ngIf=”!p.available”disabled>Out of

                        Stock</button>

                </div>

            </div>

            <divclass=”d-flex justify-content-center mt-1”>

                <pagination-controlspreviousLabel=”Prev”nextLabel=”Next”(pageChange)=”onTableDataChange($event)”>

                </pagination-controls>

            </div>

        </div>

    </div>

</div>

**Get-product.component.ts**

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

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

import{ CartItem } from‘src/app/cart-item’;

import{ Product } from‘src/app/product’;

import{ CartService } from‘src/app/Services/cart.service’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-get-product’,

  templateUrl:‘./get-product.component.html’,

  styleUrls: [‘./get-product.component.css’]

})

exportclassGetProductComponentimplementsOnInit {

  category!:string;

  product!:Product[];

  name!:string;

  page:number=1;

  count:number=0;

  tableSize:number=7;

  constructor(privateroute:ActivatedRoute, privateuserService:UserService, privatecartService:CartService, privaterouter:Router) {

  }

  ngOnInit():void {

    this.category=this.route.snapshot.params[‘category’];

    this.showMedicineByCategory();

  }

  onTableDataChange(event:any) {

    this.page=event;

  }

  showMedicineByCategory() {

    if (this.category==‘All-Medicines’) {

      this.userService.getAllMedicine().subscribe({

        next: (data) => {

          this.product=data;

          this.product.forEach((p) => {

            p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

          })

        }, error: (error) => {

          console.log(error);

          alert(‘No Medicines Found’);

        }

      })

    } else {

      this.userService.getMedicineByCategory(this.category).subscribe({

        next: (data) => {

          this.product=data;

          this.product.forEach((p) => {

            p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

          })

        }, error: (error) => {

          console.log(error);

          alert(‘No Medicines Found’);

        }

      })

    }

  }

  onSearch(name:string) {

    if (name!=undefined) {

      console.log(‘navigating to search url’);

      leturl=“/user/search/product/”+name;

      this.router.navigateByUrl(url);

    } else {

      console.log(‘please enter a name’);

    }

  }

  sortByPriceLowToHigh() {

    this.product.sort((a, b) =>a.price–b.price);

  }

  sortByPriceHighToLow() {

    this.product.sort((a, b) =>b.price–a.price);

  }

  sortByNameAscending() {

    this.product.sort((a, b) =>a.name.localeCompare(b.name));

  }

  sortByNameDescending() {

    this.product.sort((a, b) =>b.name.localeCompare(a.name));

  }

  addToCart(product:Product) {

    constcartItem=newCartItem(product);

    this.cartService.addToCart(cartItem);

  }

}

**Cart-details.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-lg-8 col-md-10 col-sm-12”>

            <h3>Items in your cart({{totalQuantity}})</h3>

            <div\*ngIf=”cartItems.length == 0”>

                <divclass=”col-6 offset-lg-6”>

                    <divclass=”alert alert-warning text-center”><strong>Your Cart is empty!</strong></div>

                </div>

            </div>

            <div\*ngIf=”cartItems.length> 0”>

                <tableclass=”table table-hover”>

                    <thead>

                        <tr>

                            <th[width]=”150”>

                                Products

                            </th>

                            <th[width]=”400”></th>

                            <th></th>

                            <th></th>

                        </tr>

                    </thead>

                    <tbody>

                        <tr\*ngFor=”let item of cartItems”>

                            <td>

                                <imgclass=”img-thumbnail”width=”150”height=”200”[src]=”item.img”>

                            </td>

                            <td>

                                <h5>{{item.name}}</h5>

                                <pclass=”text-muted”>By {{item.brand}}</p>

                                <p>MRP: <b>{{item.price | currency: ‘INR’}}\*</b></p>

                            </td>

                            <td><br><buttonclass=”btnbtn-secondary btn-sm”(click)=”increamentQuantity(item)”><i

                                        class=”fas fa-plus”></i></button>&nbsp;<b>{{item.quantity}}</b>&nbsp;<button

                                    class=”btnbtn-secondary btn-sm”(click)=”decrementQuantity(item)”><i

                                        class=”fas fa-minus”></i></button>

                            </td>

                            <td><br><buttonclass=”btnbtn-danger btn-sm”(click)=”removeItem(item)”><i

                                        class=”fas fa-trash-alt”></i></button></td>

                        </tr>

                    </tbody>

                </table>

            </div>

        </div>

        <divclass=”col-lg-4 col-md-12 ml-auto mt-5”\*ngIf=”cartItems.length> 0”>

            <divclass=”card”>

                <divclass=”card-header”>

                    <h4>Order Summary <spanclass=”text-muted”>({{totalQuantity}} item)</span></h4>

                </div>

                <divclass=”card-body text-justify mx-auto”>

                    <pclass=”card-text”>

                        Total Quantity: &nbsp;&nbsp;{{totalQuantity}}

                    </p>

                    <pclass=”card-text”>

                        Delivery Charges: <spanclass=”text-success”>&nbsp;&nbsp;Free</span>

                    </p>

                    <pclass=”card-text font-weight-bold”>

                        <spanclass=”text-info”>Total Payable: &nbsp;&nbsp;{{totalPrice | currency: ‘INR’}}</span>

                    </p>

                </div>

                <divclass=”card-footer”>

                    <arouterLink=”/user/create/order”role=”button”class=”btnbtn-success btn-block”>Proceed to Pay

                        {{totalPrice | currency:

                        ‘INR’}}</a>

                </div>

            </div>

        </div>

    </div>

</div>

**Cart-details.comonent.ts**

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

import{ CartItem } from‘src/app/cart-item’;

import{ CartService } from‘src/app/Services/cart.service’;

@Component({

  selector:‘app-cart-details’,

  templateUrl:‘./cart-details.component.html’,

  styleUrls: [‘./cart-details.component.css’]

})

exportclassCartDetailsComponentimplementsOnInit {

  constructor(privatecartService:CartService) { }

  ngOnInit():void {

    this.cartDetails();

  }

  cartItems:CartItem[] = [];

  totalPrice:number=0;

  totalQuantity:number=0;

  cartDetails() {

    this.cartItems=this.cartService.cartItems;

    this.cartService.totalPrice.subscribe(data=>this.totalPrice=data);

    this.cartService.totalQuantity.subscribe(data=>this.totalQuantity=data);

    this.cartService.calculateTotalPrice();

  }

  increamentQuantity(cartItem:CartItem) {

    this.cartService.addToCart(cartItem);

  }

  decrementQuantity(cartItem:CartItem) {

    this.cartService.decrementQuantity(cartItem);

  }

  removeItem(cartItem:CartItem) {

    this.cartService.remove(cartItem);

  }

}

**Search-product.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-sm-12 col-md-10 col-lg-8 mx-auto”>

            <br>

            <p>Search results for <b>{{medicineName}}</b></p>

            <spanstyle=”font-size: 1.5rem;”>Products</span>

            <spanclass=”float-right mt-2”><aclass=”btnbtn-light btn-sm dropdown-toggle”href=”#”

                    data-toggle=”dropdown”role=”button”>

                    Sort by

                </a>

                <divclass=”dropdown-menu”>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceLowToHigh()”>Price: Low To High</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByPriceHighToLow()”>Price: High To Low</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameAscending()”>Name: A-Z</a>

                    <aclass=”dropdown-item”role=”button”(click)=”sortByNameDescending()”>Name: Z-A</a>

                </div>

            </span>

            <divclass=”card flex-row mt-3 bg-light”\*ngFor=”let p of product | paginate :

                                                                                {

                                                                                    itemsPerPage: tableSize,

                                                                                    currentPage: page,

                                                                                    totalItems: count

                                                                                }”>

                <img[src]=”p.img”class=”card-img-left img-thumbnail”width=”200”height=”200”>

                <divclass=”card-body”>

                    <h4class=”card-title”>{{p.name}} <spanclass=”float-right”>&#8377; {{p.price}}</span></h4>

                    <pclass=”text-muted”>By {{p.brand}}</p>

                    <small>Description: {{p.description}}</small>

                    <br>

                    <small>Contains: <spanclass=”font-weight-bold”>{{p.salt}}</span></small>

                    <buttonclass=”btnbtn-primary float-right mt-2”\*ngIf=”p.available”(click)=”addToCart(p)”>Add to

                        cart</button>

                    <buttonclass=”btnbtn-secondary float-right mt-2”\*ngIf=”!p.available”disabled>Out of

                        Stock</button>

                </div>

            </div>

            <divclass=”d-flex justify-content-center mt-1”>

                <pagination-controlspreviousLabel=”Prev”nextLabel=”Next”(pageChange)=”onTableDataChange($event)”>

                </pagination-controls>

            </div>

        </div>

    </div>

</div>

**Search-product.component.ts**

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

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

import{ CartItem } from‘src/app/cart-item’;

import{ Product } from‘src/app/product’;

import{ CartService } from‘src/app/Services/cart.service’;

import{ UserService } from‘src/app/Services/user.service’;

@Component({

  selector:‘app-search-product’,

  templateUrl:‘./search-product.component.html’,

  styleUrls: [‘./search-product.component.css’]

})

exportclassSearchProductComponentimplementsOnInit {

  medicineName!:string;

  name!:string;

  product!:Product[];

  page:number=1;

  count:number=0;

  tableSize:number=7;

  constructor(privateroute:ActivatedRoute, privateuserService:UserService, privatecartService:CartService, privaterouter:Router) {

  }

  ngOnInit():void {

    this.medicineName=this.route.snapshot.params[‘name’];

    console.log(this.medicineName);

    this.getProductByName();

  }

  getProductByName() {

    this.userService.getMedicineByName(this.medicineName).subscribe({

      next: (data) => {

        this.product=data;

        this.product.forEach((p) => {

          p.img=‘data:image/jpeg;base64,’+p.productImage.imageData;

        })

      }, error: (error) => {

        console.log(error);

        alert(‘No Medicines Found’);

      }

    })

  }

  onTableDataChange(event:any) {

    this.page=event;

  }

  sortByPriceLowToHigh() {

    this.product.sort((a, b) =>a.price–b.price);

  }

  sortByPriceHighToLow() {

    this.product.sort((a, b) =>b.price–a.price);

  }

  sortByNameAscending() {

    this.product.sort((a, b) =>a.name.localeCompare(b.name));

  }

  sortByNameDescending() {

    this.product.sort((a, b) =>b.name.localeCompare(a.name));

  }

  addToCart(product:Product) {

    constcartItem=newCartItem(product);

    this.cartService.addToCart(cartItem);

  }

}

* **User:**

**User-home.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col">

            <divclass="jumbotron jumbotron-fluid bg-secondary">

                <divclass="text-center text-white">

                    <h1>Welcome {{name}} !</h1>

                </div>

            </div>

            <divclass="card-deck col-9 mx-auto mb-2">

                <divclass="card bg-success">

                    <imgclass="card-img-top"src="assets/Images/cart.png"alt="cart.png"

                        style="width:100%; height:250px;">

                    <divclass="card-body text-center">

                        <h4class="card-title">Your Cart</h4>

                        <pclass="card-text">Display all the cart items.</p>

                    </div>

                    <divclass="card-footer">

                        <arouterLink="/get/cart/details"role="button"class="btnbtn-info btn-block">Go to Cart</a>

                    </div>

                </div>

                <divclass="card bg-success">

                    <imgclass="card-img-top"src="assets/Images/order.png"alt="order.png"

                        style="width:100%; height:250px;">

                    <divclass="card-body text-center">

                        <h4class="card-title">All Orders</h4>

                        <pclass="card-text">Display all the order's placed.</p>

                    </div>

                    <divclass="card-footer">

                        <buttonclass="btnbtn-info btn-block"(click)="getOrders()">Show Orders</button>

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

**User-home.component.ts**

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

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

import{ LoginService } from'src/app/Services/login.service';

@Component({

  selector:'app-user-home',

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

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

})

exportclassUserHomeComponentimplementsOnInit {

  username!:string;

  name!:string;

  constructor(privaterouter:Router, privateloginService:LoginService) { }

  ngOnInit():void {

    this.username=this.loginService.getUserDetails().username;

    this.name=this.loginService.getUserDetails().firstName+' '+this.loginService.getUserDetails().lastName;

  }

  getOrders() {

    leturl='/user/get/all-orders/'+this.username;

    this.router.navigateByUrl(url);

  }

}

**User-login.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col-lg-4 col-md-8 mx-auto">

            <divclass="card mt-5 bg-info">

                <divclass="card-body">

                    <divclass="text-center">

                        <h4class="card-title">Welcome User</h4>

                        <p>Sign in to continue.</p>

                    </div>

                    <hr>

                    <form#login="ngForm"(ngSubmit)="onSubmit()">

                        <divclass="form-group">

                            <labelfor="email"><b>Username:</b></label><inputtype="text"name="username"

                                #username="ngModel"class="form-control"placeholder="Enter Username"id="email"

                                [(ngModel)]="credentials.username"requiredngModel>

                            <span\*ngIf="username.invalid&&username.touched"style="color:red">Please enter a valid

                                username.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="pwd"><b>Password:</b></label><inputtype="password"name="password"

                                #password="ngModel"minlength="6"class="form-control"placeholder="Enter password"

                                id="pwd"[(ngModel)]="credentials.password"requiredngModel>

                            <span\*ngIf="password.invalid&&password.touched"style="color: red">Please enter a valid

                                password.</span>

                        </div>

                        <buttontype="submit"class="btnbtn-danger btn-block"[disabled]="login.invalid">Login</button>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**User-login.component.ts**

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

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

import{ Credentials } from'src/app/credentials';

import{ LoginService } from'src/app/Services/login.service';

@Component({

  selector:'app-user-login',

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

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

})

exportclassUserLoginComponent {

  credentials:Credentials=newCredentials();

  constructor(privateloginService:LoginService, privaterouter:Router) { }

  onSubmit() {

    this.loginService.generateToken(this.credentials).subscribe({

      next: (response) => {

        //user is logged in

        this.loginService.loginUser(response.token);

        this.loginService.getCurrentUser().subscribe({

          next: (user) => {

            this.loginService.setUserDetails(user);

            //redirect: to user home page

            if (this.loginService.getUserRole() =='USER') {

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

            }

          }, error: (error) => {

            console.log(error);

          }

        })

      }, error: (error) => {

        console.log(error);

        alert('Invalid Credentials!');

      }

    })

  }

}

**User-signup.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col-lg-6 col-md-8 mx-auto mt-3 mb-4">

            <divclass="card">

                <divclass="card-header">

                    <h4class="text-center">User Registration</h4>

                </div>

                <divclass="card-body">

                    <form#registerationForm="ngForm"(ngSubmit)="onSubmit()">

                        <divclass="form-group">

                            <labelfor="username">Your E-mail:</label>

                            <inputtype="email"class="form-control"required#username="ngModel"

                                placeholder="Enter your E-mail address"id="username"name="username"

                                [(ngModel)]="user.username"ngModel>

                            <span\*ngIf="username.invalid&&username.touched"style="color: red">Please enter a valid

                                e-mail.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="pwd">Enter a Password:</label>

                            <inputtype="password"required#password="ngModel"class="form-control"

                                placeholder="Enter a password of minimum 6 characters."id="pwd"minlength="6"

                                name="password"[(ngModel)]="user.password"ngModel>

                            <span\*ngIf="password.invalid&&password.touched"style="color: red">Please enter valid

                                password of minimum 6 characters.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="fname">First Name:</label>

                            <inputtype="text"class="form-control"required#firstName="ngModel"

                                placeholder="Enter your first name"id="fname"name="firstName"

                                [(ngModel)]="user.firstName"ngModel>

                            <span\*ngIf="firstName.invalid&&firstName.touched"style="color:red">Please enter valid

                                first name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="lname">Last Name:</label>

                            <inputtype="text"required#lastName="ngModel"class="form-control"

                                placeholder="Enter your last name"id="lname"name="lastName"

                                [(ngModel)]="user.lastName"ngModel>

                            <span\*ngIf="lastName.invalid&&lastName.touched"style="color: red">Please enter valid

                                last name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="phn">Contact no. :</label>

                            <divclass="input-group">

                                <divclass="input-group-prepend">

                                    <spanclass="input-group-text">+91</span>

                                </div>

                                <inputtype="text"required#phone="ngModel"class="form-control"

                                    placeholder="Enter your phone number"id="phn"name="phone"minlength="10"

                                    [(ngModel)]="user.contactNumber"ngModel>

                            </div>

                            <span\*ngIf="phone.invalid&&phone.touched"style="color:red">Please enter valid contact

                                number</span>

                        </div>

                        <divclass="text-center">

                            <divclass="btn-group">

                                <buttontype="submit"class="btnbtn-primary"[disabled]="registerationForm.invalid"

                                    data-toggle="modal"data-target="#myModal">Sign Up</button>

                                <!-- The Modal -->

                                <divclass="modal fade"id="myModal">

                                    <divclass="modal-dialog modal-dialog-centered">

                                        <divclass="modal-content">

                                            <!-- Modal Header -->

                                            <divclass="modal-header">

                                                <h4class="modal-title">User Registration</h4>

                                                <buttontype="button"class="close"(click)="onClick()"

                                                    data-dismiss="modal">&times;</button>

                                            </div>

                                            <!-- Modal body -->

                                            <divclass="modal-body">

                                                <span\*ngIf="isValid"style="color:green">{{message}}</span>

                                                <span\*ngIf="!isValid"style="color: red">{{message}}</span>

                                            </div>

                                            <!-- Modal footer -->

                                            <divclass="modal-footer">

                                                <buttontype="button"class="btnbtn-secondary"data-dismiss="modal"

                                                    (click)="onClick()">Close</button>

                                            </div>

                                        </div>

                                    </div>

                                </div>

                                <buttontype="reset"class="btnbtn-danger">Reset</button>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**User-signup.component.ts**

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

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

import{ UserService } from'src/app/Services/user.service';

import{ User } from'src/app/user';

@Component({

  selector:'app-user-signup',

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

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

})

exportclassUserSignupComponent {

  user:User=newUser();

  isValid!:boolean;

  message!:string;

  constructor(privateuserService:UserService, privaterouter:Router) { }

  onSubmit() {

    this.userService.userSignUp(this.user).subscribe({

      next: (response) => {

        this.isValid=true;

        this.message='Successfully Registered! Sign In to Continue!';

      }, error: (error) => {

        console.log(error);

        this.isValid=false;

        this.message='E-mail address already exists!';

      }

    })

  }

  onClick() {

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

  }

}

**Order-details.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col-lg-6 col-md-8 mx-auto mt-3 mb-4">

            <divclass="card">

                <divclass="card-header">

                    <h4class="text-center">Order Details</h4>

                </div>

                <divclass="card-body">

                    <form#orderDetailForm="ngForm"(ngSubmit)="onSubmit()">

                        <divclass="form-group">

                            <labelfor="fname">Your First Name:</label>

                            <inputtype="text"class="form-control"required#fname="ngModel"

                                placeholder="Enter your your first name"id="fname"name="firstName"

                                [(ngModel)]="orderDetails.firstName"ngModel>

                            <span\*ngIf="fname.invalid&&fname.touched"style="color: red">Please enter a valid

                                first name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="lname">Your Last Name:</label>

                            <inputtype="text"required#lname="ngModel"class="form-control"

                                placeholder="Enter your last name"id="lname"name="lastName"

                                [(ngModel)]="orderDetails.lastName"ngModel>

                            <span\*ngIf="lname.invalid&&lname.touched"style="color: red">Please enter a valid

                                last name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="add">Your Shipping Address:</label>

                            <textareaclass="form-control"rows="2"id="add"name="address"

                                [(ngModel)]="orderDetails.address"#add="ngModel"

                                placeholder="Enter your shipping address"requiredngModel></textarea>

                            <span\*ngIf="add.invalid&&add.touched"style="color: red">Please enter a valid

                                shipping address.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="pc">Your Area Pincode:</label>

                            <inputtype="number"required#pc="ngModel"class="form-control"minlength="6"

                                placeholder="Enter your area pincode"id="pc"name="pinCode"

                                [(ngModel)]="orderDetails.pinCode"ngModel>

                            <span\*ngIf="pc.invalid&&pc.touched"style="color: red">Please enter a valid area

                                pincode.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="dt">Your District:</label>

                            <inputtype="text"required#dt="ngModel"class="form-control"

                                placeholder="Enter your district name"id="dt"name="district"

                                [(ngModel)]="orderDetails.district"ngModel>

                            <span\*ngIf="dt.invalid&&dt.touched"style="color: red">Please enter a valid

                                district name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="ste">Your State:</label>

                            <inputtype="text"required#ste="ngModel"class="form-control"

                                placeholder="Enter your state name"id="ste"name="state"

                                [(ngModel)]="orderDetails.state"ngModel>

                            <span\*ngIf="ste.invalid&&ste.touched"style="color: red">Please enter a valid

                                state name.</span>

                        </div>

                        <divclass="form-group">

                            <labelfor="phn">Your Contact no. :</label>

                            <divclass="input-group">

                                <divclass="input-group-prepend">

                                    <spanclass="input-group-text">+91</span>

                                </div>

                                <inputtype="text"required#phone="ngModel"class="form-control"

                                    placeholder="Enter your contact number"id="phn"name="contact"minlength="10"

                                    [(ngModel)]="orderDetails.contact"ngModel>

                            </div>

                            <span\*ngIf="phone.invalid&&phone.touched"style="color:red">Please enter valid contact

                                number</span>

                        </div>

                        <divclass="text-center">

                            <buttontype="button"class="btnbtn-success btn-lg"[disabled]="orderDetailForm.invalid"

                                data-toggle="modal"data-target="#myModal">Pay Now {{paidAmount | currency:

                                'INR'}}</button>

                        </div>

                        <!-- The Modal -->

                        <divclass="modal fade"id="myModal">

                            <divclass="modal-dialog modal-dialog-centered">

                                <divclass="modal-content">

                                    <!-- Modal Header -->

                                    <divclass="modal-header">

                                        <h4class="modal-title">Kindly enter your card details</h4>

                                        <buttontype="button"class="close"data-dismiss="modal">&times;</button>

                                    </div>

                                    <!-- Modal body -->

                                    <divclass="modal-body">

                                        <divclass="was-validated">

                                            <divclass="form-group">

                                                <labelfor="uname">Enter the name on the card:</label>

                                                <inputtype="text"class="form-control"id="uname"

                                                    placeholder="Enter the name on card"name="cname"required>

                                                <divclass="valid-feedback">Valid.</div>

                                                <divclass="invalid-feedback">Please fill out this field.

                                                </div>

                                            </div>

                                            <divclass="form-group">

                                                <labelfor="crd">Enter your card number:</label>

                                                <inputtype="number"class="form-control"id="crd"

                                                    placeholder="Enter the card number"name="cnumber"required>

                                                <divclass="valid-feedback">Valid.</div>

                                                <divclass="invalid-feedback">Please fill out this field.

                                                </div>

                                            </div>

                                            <divclass="form-group">

                                                <labelfor="cvvn">Enter your CVV number:</label>

                                                <inputtype="password"class="form-control"id="cvvn"

                                                    placeholder="Enter your cvv"name="cvv"required>

                                                <divclass="valid-feedback">Valid.</div>

                                                <divclass="invalid-feedback">Please fill out this field.

                                                </div>

                                            </div>

                                            <divclass="form-group form-check">

                                                <labelclass="form-check-label">

                                                    <inputclass="form-check-input"type="checkbox"name="remember"

                                                        required> I agree to the terms and

                                                    conditions.

                                                    <divclass="valid-feedback">Valid.</div>

                                                    <divclass="invalid-feedback">Check this checkbox to

                                                        continue.</div>

                                                </label>

                                            </div>

                                            <div>

                                                <h5class="text-center">Payable amount = {{paidAmount | currency:

                                                    'INR'}} </h5>

                                            </div>

                                            <divclass="text-center">

                                                <buttontype="submit"class="btnbtn-success">Make Payment</button>

                                            </div>

                                        </div>

                                    </div>

                                    <!-- Modal footer -->

                                    <divclass="modal-footer">

                                        <buttontype="button"class="btnbtn-danger"

                                            data-dismiss="modal">Cancel</button>

                                    </div>

                                </div>

                            </div>

                        </div>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**Order-details.component.ts**

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

import{ CartItem } from'src/app/cart-item';

import{ OrderDetails } from'src/app/order-details';

import{ OrderItem } from'src/app/order-item';

import{ CartService } from'src/app/Services/cart.service';

import{ LoginService } from'src/app/Services/login.service';

import{ UserService } from'src/app/Services/user.service';

@Component({

  selector:'app-order-details',

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

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

})

exportclassOrderDetailsComponentimplementsOnInit {

  orderDetails:OrderDetails=newOrderDetails();

  cartItems:CartItem[] = [];

  orderItem:OrderItem[] = [];

  paidAmount:number=0;

  username!:string;

  constructor(privatecartService:CartService, privateloginService:LoginService, privateuserService:UserService) { }

  ngOnInit():void {

    this.cartItems=this.cartService.cartItems;

    for (letcartItemsofthis.cartItems) {

      letitems:OrderItem=newOrderItem();

      items.pid=cartItems.pid;

      items.quantity=cartItems.quantity;

      this.orderItem.push(items);

    }

    this.cartService.totalPrice.subscribe(data=>this.paidAmount=data);

    this.username=this.loginService.getUserDetails().username;

    this.cartService.calculateTotalPrice();

    this.orderDetails.username=this.username;

    this.orderDetails.paidAmount=this.paidAmount;

    this.orderDetails.paymentMode="CARD-PAYMENT";

    this.orderDetails.cartItem=this.orderItem;

  }

  onSubmit() {

    this.userService.createOrder(this.orderDetails).subscribe({

      next: (data) => {

        window.location.href="/order-confirmation/invoice/"+data.oid;

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**Order-confirmtion.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col-8 mx-auto mt-2 mb-2">

            <divclass="card bg-light">

                <divclass="card-header">

                    <h3class="text-center font-weight-bold">Order Confirmation</h3>

                    <pclass="text-center">Dear <b>{{orderInvoice.firstName}} {{orderInvoice.lastName}}</b>, Thank you

                        for shopping with us!</p>

                </div>

                <divclass="card-body">

                    <h4class="text-center"><b>Order Summary</b></h4>

                    <pclass="text-center">{{orderInvoice.date}}</p>

                    <divclass="card mx-auto">

                        <divclass="card-body">

                            <p><b>Order Id: </b> {{orderInvoice.oid}} <spanclass="float-right"><b>Order Status:

                                    </b>{{orderInvoice.status}}</span></p>

                            <p>Ordered Products:</p>

                            <divclass="card flex-row mt-3 bg-light"\*ngFor="let p of orderInvoice.products">

                                <img[src]="p.product.img"class="card-img-left img-thumbnail"width="100"height="100">

                                <divclass="card-body">

                                    <h4class="card-title">{{p.product.name}} <span

                                            class="float-right text-muted">&#8377;

                                            {{p.product.price \* p.quantity}}</span></h4>

                                    <pclass="text-muted">By {{p.product.brand}} <span

                                            class="text-dark float-right">Quantity:

                                            {{p.quantity}}</span></p>

                                </div>

                            </div>

                            <hr>

                            <divclass="text-center">

                                <h5><b>Order Total: </b><spanclass="text-muted">{{orderInvoice.paidAmount |

                                        currency:'INR'}}</span></h5>

                                <p><b>Payment mode: </b>{{orderInvoice.paymentMode}}</p>

                                <p><b>Total Amount Paid: </b>{{orderInvoice.paidAmount | currency:'INR'}}</p>

                            </div>

                            <hr>

                            <divclass="text-center">

                                <h5><b>Shipping details</b></h5>

                                <p><b>Shipping address: </b>{{orderInvoice.address}}, {{orderInvoice.district}},

                                    {{orderInvoice.pinCode}}, {{orderInvoice.state}}.</p>

                                <p><b>Contact detail: </b>+91 {{orderInvoice.contact}}</p>

                            </div>

                        </div>

                    </div>

                </div>

                <divclass="card-footer">

                    <h6class="text-center">Your order is on it's way. Thank you!</h6>

                </div>

            </div>

        </div>

    </div>

</div>

**Order-confirmation.component.ts**

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

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

import{ OrderSummary } from'src/app/order-summary';

import{ UserService } from'src/app/Services/user.service';

@Component({

  selector:'app-order-confirmation',

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

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

})

exportclassOrderConfirmationComponentimplementsOnInit {

  oid!:number;

  orderInvoice:OrderSummary=newOrderSummary();

  constructor(privateroute:ActivatedRoute, privateuserService:UserService) { }

  ngOnInit():void {

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

    this.getOrderConfirmation();

  }

  getOrderConfirmation() {

    this.userService.getOrderById(this.oid).subscribe({

      next: (data) => {

        this.orderInvoice=data;

        this.orderInvoice.products.forEach((p) => {

          p.product.img='data:image/jpeg;base64,'+p.product.productImage.imageData;

        })

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**All-user-orders.component.html**

<divclass="container">

    <divclass="row">

        <divclass="col mt-2">

            <divclass="table-responsive">

                <tableclass="table table-bordered">

                    <theadclass="thead-dark text-center">

                        <tr>

                            <th>S.No.</th>

                            <th>Order-Date</th>

                            <th>Order-Id</th>

                            <th>Paid Amount</th>

                            <th>Order-Details</th>

                        </tr>

                    </thead>

                    <tbody>

                        <trclass="table-secondary text-center"\*ngFor=" let o of orders">

                            <td></td>

                            <td>{{o.date}}</td>

                            <td>{{o.oid}}</td>

                            <td>{{o.paidAmount | currency: 'INR'}}</td>

                            <td>

                                <buttonclass="btnbtn-success"(click)="getOrderDetails(o.oid)">View Order

                                    Details</button>

                            </td>

                        </tr>

                    </tbody>

                </table>

            </div>

        </div>

    </div>

</div>

**All-user-order-component.ts**

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

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

import{ UserService } from'src/app/Services/user.service';

@Component({

  selector:'app-all-user-orders',

  templateUrl:'./all-user-orders.component.html',

  styleUrls: ['./all-user-orders.component.css']

})

exportclassAllUserOrdersComponentimplementsOnInit {

  username!:string;

  orders:any[] = [];

  constructor(privateroute:ActivatedRoute, privateuserService:UserService, privaterouter:Router) { }

  ngOnInit():void {

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

    this.getAllOrders();

  }

  getAllOrders() {

    this.userService.getOrderByUsername(this.username).subscribe({

      next: (data) => {

        this.orders=data;

      }, error: (error) => {

        console.log(error);

        alert('No Orders found');

      }

    })

  }

  getOrderDetails(oid:number) {

    leturl='/order-confirmation/invoice/'+oid;

    this.router.navigateByUrl(url);

  }

}

**Backend-SpringBoot**

1. **Controller:**

**JwtController.java**

packagecom.medicare.controller;

importjava.security.Principal;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.http.ResponseEntity;

importorg.springframework.security.authentication.AuthenticationManager;

importorg.springframework.security.authentication.BadCredentialsException;

importorg.springframework.security.authentication.DisabledException;

importorg.springframework.security.authentication.UsernamePasswordAuthenticationToken;

importorg.springframework.security.core.userdetails.UserDetails;

importorg.springframework.security.core.userdetails.UsernameNotFoundException;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.GetMapping;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RestController;

importcom.medicare.config.JwtUtil;

importcom.medicare.entities.JwtRequest;

importcom.medicare.entities.JwtResponse;

importcom.medicare.entities.User;

importcom.medicare.services.UserDetailService;

@RestController

@CrossOrigin(origins ="\*")

publicclassJwtController {

    @Autowired

    privateAuthenticationManagerauthenticationManager;

    @Autowired

    privateUserDetailServiceuserDetailService;

    @Autowired

    privateJwtUtiljwtUtil;

    //generate token

    @PostMapping("/generate-token")

    publicResponseEntity<?>generateToken(@RequestBodyJwtRequestjwtRequest) throwsException{

        try {

            authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());

        }catch(UsernameNotFoundExceptione) {

            e.printStackTrace();

            thrownewException("User does not exist or invalid credentials!");

        }

        // validated

        UserDetailsuserDetails=this.userDetailService.loadUserByUsername(jwtRequest.getUsername());

        Stringtoken=this.jwtUtil.generateToken(userDetails);

        returnResponseEntity.ok(newJwtResponse(token));

    }

    privatevoidauthenticate(Stringusername, Stringpassword) throwsException {

        try {

            this.authenticationManager.authenticate(newUsernamePasswordAuthenticationToken(username, password));

        } catch (BadCredentialsExceptione) {

            thrownewException("Invalid Credentials! "+e.getMessage());

        }catch(DisabledExceptione) {

            thrownewException("User Disabled! "+e.getMessage());

        }

    }

    @GetMapping("/current-user")

    publicUsergetCurrentUser(Principalprincipal) {

        return (User)this.userDetailService.loadUserByUsername(principal.getName());

    }

}

**ProductController.java**

packagecom.medicare.controller;

importjava.io.IOException;

importjava.util.List;

importjavax.validation.Valid;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

importorg.springframework.security.access.prepost.PreAuthorize;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.DeleteMapping;

importorg.springframework.web.bind.annotation.GetMapping;

importorg.springframework.web.bind.annotation.PathVariable;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.PutMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RequestParam;

importorg.springframework.web.bind.annotation.RestController;

importorg.springframework.web.multipart.MultipartFile;

importcom.fasterxml.jackson.core.JsonProcessingException;

importcom.fasterxml.jackson.databind.JsonMappingException;

importcom.fasterxml.jackson.databind.ObjectMapper;

importcom.medicare.config.ImageUtil;

importcom.medicare.entities.Product;

importcom.medicare.entities.ProductImage;

importcom.medicare.services.ProductService;

@RestController

@CrossOrigin(origins ="\*")

publicclassProductController {

    @Autowired

    privateProductServiceproductService;

    @Autowired

    privateObjectMapperobjectMapper;

    //add new product

    @PreAuthorize("hasAuthority('ADMIN')")

    @PostMapping("/add/product")

    publicResponseEntity<?>addNewProduct(@RequestParam("product") Stringproduct,

                                           @RequestParam("image") MultipartFilefile) throwsIOException{

        ProductImageimg=newProductImage();

        img.setName(file.getOriginalFilename());

        img.setType(file.getContentType());

        img.setImageData(ImageUtil.compressImage(file.getBytes()));

        Productp=null;

        try {

            p =objectMapper.readValue(product,Product.class);

            p.setProductImage(img);

        } catch (JsonMappingExceptione) {

            e.printStackTrace();

        } catch (JsonProcessingExceptione) {

            e.printStackTrace();

            returnResponseEntity.status(HttpStatus.BAD\_REQUEST).body("Invalid Request");

        }

        ProductsaveProduct=this.productService.addProduct(p);

        returnResponseEntity.ok(saveProduct);

    }

    //update existing product

    @PreAuthorize("hasAuthority('ADMIN')")

    @PutMapping("/update/product/{id}")

    publicResponseEntity<?>updateProduct(@PathVariable("id") Longid,@Valid @RequestBodyProductproduct){

        ProductupdateProduct=this.productService.findProduct(id);

        updateProduct.setName(product.getName());

        updateProduct.setBrand(product.getBrand());

        updateProduct.setCategory(product.getCategory());

        updateProduct.setDescription(product.getDescription());

        updateProduct.setSalt(product.getSalt());

        updateProduct.setTotalAvailable(product.getTotalAvailable());

        updateProduct.setPrice(product.getPrice());

        this.productService.addProduct(updateProduct);

        returnResponseEntity.status(HttpStatus.CREATED).build();

    }

    //find product by id

    @GetMapping("get-product/{id}")

    publicResponseEntity<?>getProductById(@PathVariable("id") Longid){

        Productproduct=this.productService.findProduct(id);

        ProductImageimg=  newProductImage();

        img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

        img.setImgId(product.getProductImage().getImgId());

        img.setName(product.getProductImage().getName());

        img.setType(product.getProductImage().getType());

        product.setProductImage(img);

        returnResponseEntity.ok(product);

    }

    //find all products

    @GetMapping("/get/all-products")

    publicResponseEntity<?>getAllProducts(){

        List<Product>allProducts=this.productService.findAllProducts();

        allProducts.forEach(product -> {

            ProductImageimg=  newProductImage();

            img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

            img.setImgId(product.getProductImage().getImgId());

            img.setName(product.getProductImage().getName());

            img.setType(product.getProductImage().getType());

            product.setProductImage(img);

        });

        if(allProducts.isEmpty()) {

            returnResponseEntity.status(HttpStatus.NOT\_FOUND).build();

        }else {

            returnResponseEntity.ok(allProducts);

        }

    }

    @GetMapping(value = {"/get/products/{name}"})

    publicResponseEntity<?>getProductByName(@PathVariable("name") Stringname,@PathVariable("name") Stringsalt){

        List<Product>products=this.productService.findByNameOrSalt(name, salt);

        products.forEach(product -> {

            ProductImageimg=  newProductImage();

            img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

            img.setImgId(product.getProductImage().getImgId());

            img.setName(product.getProductImage().getName());

            img.setType(product.getProductImage().getType());

            product.setProductImage(img);

        });

        if(products.isEmpty()) {

            returnResponseEntity.status(HttpStatus.NOT\_FOUND).build();

        }else {

            returnResponseEntity.ok(products);

        }

    }

    @GetMapping("/get/products-by-category/{category}")

    publicResponseEntity<?>getProductsByCategory(@PathVariable("category") Stringcategory){

        List<Product>products=this.productService.findProductByCategory(category);

        products.forEach(product -> {

            ProductImageimg=  newProductImage();

            img.setImageData(ImageUtil.decompressImage(product.getProductImage().getImageData()));

            img.setImgId(product.getProductImage().getImgId());

            img.setName(product.getProductImage().getName());

            img.setType(product.getProductImage().getType());

            product.setProductImage(img);

        });

        if(products.isEmpty()) {

            returnResponseEntity.status(HttpStatus.NOT\_FOUND).build();

        }else {

            returnResponseEntity.ok(products);

        }

    }

    @PreAuthorize("hasAuthority('ADMIN')")

    @DeleteMapping("/delete/product/{id}")

    publicResponseEntity<?>deleteProduct(@PathVariable("id") Longid){

        this.productService.deleteProductById(id);

        returnResponseEntity.status(HttpStatus.OK).build();

    }

    @PreAuthorize("hasAuthority('ADMIN')")

    @PutMapping("/set-availability/product/{id}")

    publicResponseEntity<?>setAvailability(@PathVariable("id") Longid, @RequestBodyProductproduct){

        ProductupdateProduct=this.productService.findProduct(id);

        updateProduct.setAvailable(product.isAvailable());

        this.productService.addProduct(updateProduct);

        returnResponseEntity.status(HttpStatus.CREATED).build();

    }

    @GetMapping("/get/{name}")

    publicResponseEntity<?>getAvailable(@PathVariable("name") Stringname){

        List<Product>products=this.productService.findTrueProduct(name);

        if(products.isEmpty()) {

            returnResponseEntity.status(HttpStatus.NOT\_FOUND).build();

        }else {

            returnResponseEntity.ok(products);

        }

    }

}

**UserController.java**

packagecom.medicare.controller;

importjava.net.URI;

importjava.util.HashSet;

importjava.util.Set;

importjavax.annotation.PostConstruct;

importjavax.validation.Valid;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RestController;

importorg.springframework.web.servlet.support.ServletUriComponentsBuilder;

importcom.medicare.entities.Role;

importcom.medicare.entities.User;

importcom.medicare.entities.UserRole;

importcom.medicare.services.UserService;

@RestController

@CrossOrigin(origins ="\*")

publicclassUserController {

    @Autowired

    privateUserServiceuserService;

    //init admin user

    @PostConstruct

    publicvoidcreateAdmin(){

        Useradmin=newUser();

        admin.setUsername("admin@medicare.com");

        admin.setPassword("admin12345");

        admin.setFirstName("Twarit");

        admin.setLastName("Soni");

        admin.setContactNumber("6265989908");

        Rolerole=newRole();

        role.setRoleId(101L);

        role.setRoleName("ADMIN");

        UserRoleur=newUserRole();

        ur.setUser(admin);

        ur.setRole(role);

        Set<UserRole>userRole=newHashSet<>();

        userRole.add(ur);

        UseradminCreated=this.userService.createUser(admin, userRole);

        System.out.println("Admin username: "+adminCreated.getUsername());

    }

    //create new user

    @PostMapping("/user/signup")

    publicResponseEntity<?>createNewUser(@Valid @RequestBodyUseruser){

        Rolerole=newRole();

        role.setRoleId(102L);

        role.setRoleName("USER");

        UserRoleur=newUserRole();

        ur.setUser(user);

        ur.setRole(role);

        Set<UserRole>userRole=newHashSet<>();

        userRole.add(ur);

        if(this.userService.getByUsername(user.getUsername())!=null) {

            System.out.println("Username already exists!");

            returnResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

        }else {

            UsernewUser=this.userService.createUser(user, userRole);

            URIlocation=ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(newUser.getUserId()).toUri();

            returnResponseEntity.created(location).build();

        }

    }

}

**UserOrderController.java**

packagecom.medicare.controller;

importjava.text.DateFormat;

importjava.util.Calendar;

importjava.util.HashSet;

importjava.util.List;

importjava.util.Set;

importjavax.validation.Valid;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

importorg.springframework.security.access.prepost.PreAuthorize;

importorg.springframework.web.bind.annotation.CrossOrigin;

importorg.springframework.web.bind.annotation.DeleteMapping;

importorg.springframework.web.bind.annotation.GetMapping;

importorg.springframework.web.bind.annotation.PathVariable;

importorg.springframework.web.bind.annotation.PostMapping;

importorg.springframework.web.bind.annotation.RequestBody;

importorg.springframework.web.bind.annotation.RestController;

importcom.medicare.config.ImageUtil;

importcom.medicare.entities.CartItem;

importcom.medicare.entities.CartOrder;

importcom.medicare.entities.Product;

importcom.medicare.entities.ProductImage;

importcom.medicare.entities.ProductQuantity;

importcom.medicare.entities.UserOrder;

importcom.medicare.services.ProductService;

importcom.medicare.services.UserOrderService;

@RestController

@CrossOrigin(origins ="\*")

publicclassUserOrderController {

    @Autowired

    privateUserOrderServiceuserOrderService;

    @Autowired

    privateProductServiceproductService;

    @PreAuthorize("hasAuthority('USER')")

    @PostMapping("/user/create/order")

    publicResponseEntity<?>createOrder(@Valid @RequestBodyCartOrdercartOrder){

        UserOrderuserOrder=newUserOrder();

        userOrder.setUsername(cartOrder.getUsername());

        userOrder.setFirstName(cartOrder.getFirstName());

        userOrder.setLastName(cartOrder.getLastName());

        userOrder.setAddress(cartOrder.getAddress());

        userOrder.setDistrict(cartOrder.getDistrict());

        userOrder.setState(cartOrder.getState());

        userOrder.setContact(cartOrder.getContact());

        userOrder.setPinCode(cartOrder.getPinCode());

        DateFormatdf=DateFormat.getDateInstance();

        Calendarcl=Calendar.getInstance();

        StringorderDate=df.format(cl.getTime());

        userOrder.setDate(orderDate);

        userOrder.setStatus("PLACED");

        userOrder.setPaidAmount(cartOrder.getPaidAmount());

        userOrder.setPaymentMode(cartOrder.getPaymentMode());

        Set<CartItem>cartItems  =cartOrder.getCartItem();

        Set<ProductQuantity>pq=newHashSet<>();

        for(CartItemitem:cartItems) {

            Productproduct=this.productService.findProduct(item.getPid());

            intquantity=item.getQuantity();

            ProductQuantityproductQuantity=newProductQuantity();

            productQuantity.setProduct(product);

            productQuantity.setQuantity(quantity);

            this.userOrderService.saveProductQuantity(productQuantity);

            pq.add(productQuantity);

        }

        userOrder.setProducts(pq);

        UserOrderorderCreated=this.userOrderService.saveOrder(userOrder);

        returnResponseEntity.ok(orderCreated);

    }

    @PreAuthorize("hasAuthority('ADMIN')")

    @GetMapping("/get/all/orders")

    publicResponseEntity<?>getAllOrders(){

        List<UserOrder>orders=this.userOrderService.getAll();

        returnResponseEntity.ok(orders);

    }

    @PreAuthorize("hasAuthority('USER')")

    @GetMapping("/get/orders/{username}")

    publicResponseEntity<?>userOrders(@PathVariable("username") Stringusername){

        List<UserOrder>orders=this.userOrderService.getUserOrders(username);

        if(orders.isEmpty()) {

            returnResponseEntity.status(HttpStatus.NOT\_FOUND).build();

        }else {

            returnResponseEntity.ok(orders);

        }

    }

    @PreAuthorize("hasAuthority('USER') or hasAuthority('ADMIN')")

    @GetMapping("/get/order-invoice/{oid}")

    publicResponseEntity<?>getUserOrderById(@PathVariable("oid") Longoid){

        UserOrderorder=this.userOrderService.getOrderById(oid);

        Set<ProductQuantity>products=order.getProducts();

        products.forEach(p -> {

            ProductImageimg=newProductImage();

            img.setImageData(ImageUtil.decompressImage(p.getProduct().getProductImage().getImageData()));

            img.setName(p.getProduct().getProductImage().getName());

            img.setImgId(p.getProduct().getProductImage().getImgId());

            img.setType(p.getProduct().getProductImage().getType());

            p.getProduct().setProductImage(img);

        });

        order.setProducts(products);

        returnResponseEntity.ok(order);

    }

    @PreAuthorize("hasAuthority('ADMIN')")

    @DeleteMapping("/delete/order/{oid}")

    publicResponseEntity<?>deleteOrderById(@PathVariable("oid") Longoid){

        this.userOrderService.deleteOrder(oid);

        returnResponseEntity.status(HttpStatus.OK).build();

    }

}

1. **Entities:**

**Authority.java**

packagecom.medicare.entities;

importorg.springframework.security.core.GrantedAuthority;

publicclassAuthorityimplementsGrantedAuthority{

    /\*\*

     \*

     \*/

    privatestaticfinallongserialVersionUID=1L;

    privateStringauthority;

    publicAuthority(Stringauthority) {

        super();

        this.authority= authority;

    }

    @Override

    publicStringgetAuthority() {

        returnthis.authority;

    }

}

**Cartitem.java**

packagecom.medicare.entities;

publicclassCartItem {

    privateLongpid;

    privateintquantity;

    publicCartItem() {

    }

    publicCartItem(Longpid, intquantity) {

        super();

        this.pid=pid;

        this.quantity= quantity;

    }

    publicLonggetPid() {

        returnpid;

    }

    publicvoidsetPid(Longpid) {

        this.pid=pid;

    }

    publicintgetQuantity() {

        return quantity;

    }

    publicvoidsetQuantity(intquantity) {

        this.quantity= quantity;

    }

}

**Cartorder.java**

packagecom.medicare.entities;

importjava.util.HashSet;

importjava.util.Set;

importjavax.validation.constraints.NotBlank;

importjavax.validation.constraints.NotEmpty;

importjavax.validation.constraints.NotNull;

publicclassCartOrder {

    @NotBlank

    privateStringusername;

    @NotBlank

    privateStringfirstName;

    @NotBlank

    privateStringlastName;

    @NotBlank

    privateStringaddress;

    @NotBlank

    privateStringdistrict;

    @NotNull

    privateintpinCode;

    @NotBlank

    privateStringstate;

    @NotBlank

    privateStringcontact;

    @NotNull

    privateDoublepaidAmount;

    @NotBlank

    privateStringpaymentMode;

    @NotEmpty

    privateSet<CartItem>cartItem=newHashSet<>();

    publicCartOrder() {

    }

    publicCartOrder(Stringusername, StringfirstName, StringlastName, Stringaddress, Stringdistrict, intpinCode,

            Stringstate, Stringcontact, DoublepaidAmount, StringpaymentMode, Set<CartItem>cartItem) {

        super();

        this.username= username;

        this.firstName=firstName;

        this.lastName=lastName;

        this.address= address;

        this.district= district;

        this.pinCode=pinCode;

        this.state= state;

        this.contact= contact;

        this.paidAmount=paidAmount;

        this.paymentMode=paymentMode;

        this.cartItem=cartItem;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetFirstName() {

        returnfirstName;

    }

    publicvoidsetFirstName(StringfirstName) {

        this.firstName=firstName;

    }

    publicStringgetLastName() {

        returnlastName;

    }

    publicvoidsetLastName(StringlastName) {

        this.lastName=lastName;

    }

    publicStringgetAddress() {

        return address;

    }

    publicvoidsetAddress(Stringaddress) {

        this.address= address;

    }

    publicStringgetDistrict() {

        return district;

    }

    publicvoidsetDistrict(Stringdistrict) {

        this.district= district;

    }

    publicintgetPinCode() {

        returnpinCode;

    }

    publicvoidsetPinCode(intpinCode) {

        this.pinCode=pinCode;

    }

    publicStringgetState() {

        return state;

    }

    publicvoidsetState(Stringstate) {

        this.state= state;

    }

    publicStringgetContact() {

        return contact;

    }

    publicvoidsetContact(Stringcontact) {

        this.contact= contact;

    }

    publicSet<CartItem>getCartItem() {

        returncartItem;

    }

    publicvoidsetCartItem(Set<CartItem>cartItem) {

        this.cartItem=cartItem;

    }

    publicDoublegetPaidAmount() {

        returnpaidAmount;

    }

    publicvoidsetPaidAmount(DoublepaidAmount) {

        this.paidAmount=paidAmount;

    }

    publicStringgetPaymentMode() {

        returnpaymentMode;

    }

    publicvoidsetPaymentMode(StringpaymentMode) {

        this.paymentMode=paymentMode;

    }

}

**Jwtrequest.java**

packagecom.medicare.entities;

publicclassJwtRequest {

    Stringusername;

    Stringpassword;

    publicJwtRequest() {

    }

    publicJwtRequest(Stringusername, Stringpassword) {

        super();

        this.username= username;

        this.password= password;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetPassword() {

        return password;

    }

    publicvoidsetPassword(Stringpassword) {

        this.password= password;

    }

}

**Jwtresponse.java**

packagecom.medicare.entities;

publicclassJwtResponse {

    Stringtoken;

    publicJwtResponse() {

    }

    publicJwtResponse(Stringtoken) {

        super();

        this.token= token;

    }

    publicStringgetToken() {

        return token;

    }

    publicvoidsetToken(Stringtoken) {

        this.token= token;

    }

}

**Product.java**

packagecom.medicare.entities;

importjavax.persistence.CascadeType;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.OneToOne;

importjavax.persistence.Table;

importjavax.validation.constraints.NotBlank;

importjavax.validation.constraints.NotNull;

importcom.fasterxml.jackson.annotation.JsonManagedReference;

@Entity

@Table(name="products")

publicclassProduct {

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLongpid;

    @NotBlank(message ="name cannot be blank")

    privateStringname;

    @NotBlank(message ="brand cannot be blank")

    privateStringbrand;

    @NotBlank(message ="category cannot be blank")

    privateStringcategory;

    @NotBlank(message ="description cannot be blank")

    privateStringdescription;

    @NotBlank(message ="salt cannot be blank")

    privateStringsalt;

    @NotNull(message ="available cannot be null")

    privateinttotalAvailable;

    @NotNull(message ="price cannot be null")

    privateDoubleprice;

    @NotNull(message ="isAvailable cannot be null")

    privatebooleanisAvailable;

    @OneToOne(cascade =CascadeType.ALL)

    @JsonManagedReference

    privateProductImageproductImage;

    publicProduct() {

        super();

    }

    publicProduct(Longpid, Stringname, Stringbrand, Stringcategory,Stringdescription, Stringsalt, inttotalAvailable, Doubleprice,

            booleanisAvailable, ProductImageproductImage) {

        super();

        this.pid=pid;

        this.name= name;

        this.brand= brand;

        this.category= category;

        this.description= description;

        this.salt= salt;

        this.totalAvailable=totalAvailable;

        this.price= price;

        this.isAvailable=isAvailable;

        this.productImage=productImage;

    }

    publicLonggetPid() {

        returnpid;

    }

    publicvoidsetPid(Longpid) {

        this.pid=pid;

    }

    publicStringgetName() {

        return name;

    }

    publicvoidsetName(Stringname) {

        this.name= name;

    }

    publicStringgetBrand() {

        return brand;

    }

    publicvoidsetBrand(Stringbrand) {

        this.brand= brand;

    }

    publicStringgetCategory() {

        return category;

    }

    publicvoidsetCategory(Stringcategory) {

        this.category= category;

    }

    publicStringgetSalt() {

        return salt;

    }

    publicvoidsetSalt(Stringsalt) {

        this.salt= salt;

    }

    publicintgetTotalAvailable() {

        returntotalAvailable;

    }

    publicvoidsetTotalAvailable(inttotalAvailable) {

        this.totalAvailable=totalAvailable;

    }

    publicDoublegetPrice() {

        return price;

    }

    publicvoidsetPrice(Doubleprice) {

        this.price= price;

    }

    publicbooleanisAvailable() {

        returnisAvailable;

    }

    publicvoidsetAvailable(booleanisAvailable) {

        this.isAvailable=isAvailable;

    }

    publicProductImagegetProductImage() {

        returnproductImage;

    }

    publicvoidsetProductImage(ProductImageproductImage) {

        this.productImage=productImage;

    }

    publicStringgetDescription() {

        return description;

    }

    publicvoidsetDescription(Stringdescription) {

        this.description= description;

    }

}

**Productimage.java**

packagecom.medicare.entities;

importjavax.persistence.Column;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.Lob;

importjavax.persistence.OneToOne;

importcom.fasterxml.jackson.annotation.JsonBackReference;

@Entity

publicclassProductImage {

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLongimgId;

    privateStringname;

    privateStringtype;

    @Lob

    @Column(name ="imagedata")

    privatebyte[] imageData;

    @OneToOne(mappedBy ="productImage")

    @JsonBackReference

    privateProductproduct;

    publicProductImage() {

        super();

    }

    publicProductImage(LongimgId, Stringname, Stringtype, byte[] imageData, Productproduct) {

        super();

        this.imgId=imgId;

        this.name= name;

        this.type= type;

        this.imageData=imageData;

        this.product= product;

    }

    publicLonggetImgId() {

        returnimgId;

    }

    publicvoidsetImgId(LongimgId) {

        this.imgId=imgId;

    }

    publicStringgetName() {

        return name;

    }

    publicvoidsetName(Stringname) {

        this.name= name;

    }

    publicStringgetType() {

        return type;

    }

    publicvoidsetType(Stringtype) {

        this.type= type;

    }

    publicbyte[] getImageData() {

        returnimageData;

    }

    publicvoidsetImageData(byte[] imageData) {

        this.imageData=imageData;

    }

    publicProductgetProduct() {

        return product;

    }

    publicvoidsetProduct(Productproduct) {

        this.product= product;

    }

}

**Productquantity.java**

packagecom.medicare.entities;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.OneToOne;

@Entity

publicclassProductQuantity {

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLongpqid;

    @OneToOne

    privateProductproduct;

    privateintquantity;

    publicProductQuantity() {

    }

    publicProductQuantity(Longpqid, Productproduct, intquantity) {

        super();

        this.pqid=pqid;

        this.product= product;

        this.quantity= quantity;

    }

    publicLonggetPqid() {

        returnpqid;

    }

    publicvoidsetPqid(Longpqid) {

        this.pqid=pqid;

    }

    publicProductgetProduct() {

        return product;

    }

    publicvoidsetProduct(Productproduct) {

        this.product= product;

    }

    publicintgetQuantity() {

        return quantity;

    }

    publicvoidsetQuantity(intquantity) {

        this.quantity= quantity;

    }

}

**Role.java**

packagecom.medicare.entities;

importjava.util.HashSet;

importjava.util.Set;

importjavax.persistence.CascadeType;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.Id;

importjavax.persistence.OneToMany;

importjavax.persistence.Table;

importjavax.validation.constraints.NotBlank;

@Entity

@Table(name ="roles")

publicclassRole {

    @Id

    privateLongroleId;

    @NotBlank(message ="role name cannot be null.")

    privateStringroleName;

    @OneToMany(cascade =CascadeType.ALL, fetch =FetchType.LAZY, mappedBy="role")

    privateSet<UserRole>userRoles=newHashSet<>();

    publicRole() {

        super();

        // TODO Auto-generated constructor stub

    }

    publicRole(LongroleId, StringroleName, Set<UserRole>userRoles) {

        super();

        this.roleId=roleId;

        this.roleName=roleName;

        this.userRoles=userRoles;

    }

    publicLonggetRoleId() {

        returnroleId;

    }

    publicvoidsetRoleId(LongroleId) {

        this.roleId=roleId;

    }

    publicStringgetRoleName() {

        returnroleName;

    }

    publicvoidsetRoleName(StringroleName) {

        this.roleName=roleName;

    }

    publicSet<UserRole>getUserRoles() {

        returnuserRoles;

    }

    publicvoidsetUserRoles(Set<UserRole>userRoles) {

        this.userRoles=userRoles;

    }

}

**User.java**

packagecom.medicare.entities;

importjava.util.Collection;

importjava.util.HashSet;

importjava.util.Set;

importjavax.persistence.CascadeType;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.OneToMany;

importjavax.persistence.Table;

importjavax.validation.constraints.NotBlank;

importjavax.validation.constraints.Size;

importorg.springframework.security.core.GrantedAuthority;

importorg.springframework.security.core.userdetails.UserDetails;

importcom.fasterxml.jackson.annotation.JsonIgnore;

@Entity

@Table(name ="users")

publicclassUserimplementsUserDetails{

    /\*\*

     \*

     \*/

    privatestaticfinallongserialVersionUID=1L;

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLonguserId;

    @NotBlank(message ="username cannot be null.")

    privateStringusername;

    @NotBlank(message ="password cannot be null.")

    @Size(min =6, message ="enter minimum six character password")

    privateStringpassword;

    @NotBlank(message ="first name cannot be null.")

    privateStringfirstName;

    @NotBlank(message ="last name cannot be null")

    privateStringlastName;

    @NotBlank(message ="contact number cannot be null")

    privateStringcontactNumber;

    privatebooleanenabled=true;

    @OneToMany(cascade =CascadeType.ALL, fetch =FetchType.EAGER, mappedBy="user")

    @JsonIgnore

    privateSet<UserRole>userRoles=newHashSet<>();

    publicUser() {

        super();

    }

    publicUser(LonguserId, Stringusername, Stringpassword, StringfirstName, StringlastName, StringcontactNumber,

            booleanenabled, Set<UserRole>userRoles) {

        super();

        this.userId=userId;

        this.username= username;

        this.password= password;

        this.firstName=firstName;

        this.lastName=lastName;

        this.contactNumber=contactNumber;

        this.enabled= enabled;

        this.userRoles=userRoles;

    }

    publicLonggetUserId() {

        returnuserId;

    }

    publicvoidsetUserId(LonguserId) {

        this.userId=userId;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetPassword() {

        return password;

    }

    publicvoidsetPassword(Stringpassword) {

        this.password= password;

    }

    publicStringgetFirstName() {

        returnfirstName;

    }

    publicvoidsetFirstName(StringfirstName) {

        this.firstName=firstName;

    }

    publicStringgetLastName() {

        returnlastName;

    }

    publicvoidsetLastName(StringlastName) {

        this.lastName=lastName;

    }

    publicStringgetContactNumber() {

        returncontactNumber;

    }

    publicvoidsetContactNumber(StringcontactNumber) {

        this.contactNumber=contactNumber;

    }

    publicbooleanisEnabled() {

        return enabled;

    }

    publicvoidsetEnabled(booleanenabled) {

        this.enabled= enabled;

    }

    publicSet<UserRole>getUserRoles() {

        returnuserRoles;

    }

    publicvoidsetUserRoles(Set<UserRole>userRoles) {

        this.userRoles=userRoles;

    }

    @Override

    publicCollection<?extendsGrantedAuthority>getAuthorities() {

        Set<Authority>authority=newHashSet<>();

        this.userRoles.forEach(userRole-> {

            authority.add(newAuthority(userRole.getRole().getRoleName()));

        });

        return authority;

    }

    @Override

    publicbooleanisAccountNonExpired() {

        // TODO Auto-generated method stub

        returntrue;

    }

    @Override

    publicbooleanisAccountNonLocked() {

        // TODO Auto-generated method stub

        returntrue;

    }

    @Override

    publicbooleanisCredentialsNonExpired() {

        // TODO Auto-generated method stub

        returntrue;

    }

}

**Useroder.java**

packagecom.medicare.entities;

importjava.util.HashSet;

importjava.util.Set;

importjavax.persistence.CascadeType;

importjavax.persistence.Entity;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.ManyToMany;

@Entity

publicclassUserOrder {

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLongoid;

    privateStringusername;

    privateStringfirstName;

    privateStringlastName;

    privateStringaddress;

    privateStringdistrict;

    privateintpinCode;

    privateStringstate;

    privateStringcontact;

    privateStringdate;

    privateStringstatus;

    privateDoublepaidAmount;

    privateStringpaymentMode;

    @ManyToMany(cascade =CascadeType.ALL)

    privateSet<ProductQuantity>products=newHashSet<>();

    publicUserOrder() {

    }

    publicUserOrder(Longoid, Stringusername, StringfirstName, StringlastName, Stringaddress, Stringdistrict,

            intpinCode, Stringstate, Stringcontact, Stringdate, Stringstatus, DoublepaidAmount, StringpaymentMode,

            Set<ProductQuantity>products) {

        super();

        this.oid=oid;

        this.username= username;

        this.firstName=firstName;

        this.lastName=lastName;

        this.address= address;

        this.district= district;

        this.pinCode=pinCode;

        this.state= state;

        this.contact= contact;

        this.date= date;

        this.status= status;

        this.paidAmount=paidAmount;

        this.paymentMode=paymentMode;

        this.products= products;

    }

    publicLonggetOid() {

        returnoid;

    }

    publicvoidsetOid(Longoid) {

        this.oid=oid;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetFirstName() {

        returnfirstName;

    }

    publicvoidsetFirstName(StringfirstName) {

        this.firstName=firstName;

    }

    publicStringgetLastName() {

        returnlastName;

    }

    publicvoidsetLastName(StringlastName) {

        this.lastName=lastName;

    }

    publicStringgetAddress() {

        return address;

    }

    publicvoidsetAddress(Stringaddress) {

        this.address= address;

    }

    publicStringgetDistrict() {

        return district;

    }

    publicvoidsetDistrict(Stringdistrict) {

        this.district= district;

    }

    publicintgetPinCode() {

        returnpinCode;

    }

    publicvoidsetPinCode(intpinCode) {

        this.pinCode=pinCode;

    }

    publicStringgetState() {

        return state;

    }

    publicvoidsetState(Stringstate) {

        this.state= state;

    }

    publicStringgetContact() {

        return contact;

    }

    publicvoidsetContact(Stringcontact) {

        this.contact= contact;

    }

    publicSet<ProductQuantity>getProducts() {

        return products;

    }

    publicvoidsetProducts(Set<ProductQuantity>products) {

        this.products= products;

    }

    publicStringgetDate() {

        return date;

    }

    publicvoidsetDate(Stringdate) {

        this.date= date;

    }

    publicStringgetStatus() {

        return status;

    }

    publicvoidsetStatus(Stringstatus) {

        this.status= status;

    }

    publicDoublegetPaidAmount() {

        returnpaidAmount;

    }

    publicvoidsetPaidAmount(DoublepaidAmount) {

        this.paidAmount=paidAmount;

    }

    publicStringgetPaymentMode() {

        returnpaymentMode;

    }

    publicvoidsetPaymentMode(StringpaymentMode) {

        this.paymentMode=paymentMode;

    }

}

**Userrole.java**

packagecom.medicare.entities;

importjavax.persistence.Entity;

importjavax.persistence.FetchType;

importjavax.persistence.GeneratedValue;

importjavax.persistence.GenerationType;

importjavax.persistence.Id;

importjavax.persistence.ManyToOne;

@Entity

publicclassUserRole {

    @Id

    @GeneratedValue(strategy =GenerationType.AUTO)

    privateLonguserRoleId;

    @ManyToOne(fetch =FetchType.EAGER)

    privateUseruser;

    @ManyToOne

    privateRolerole;

    publicUserRole() {

        super();

    }

    publicUserRole(LonguserRoleId, Useruser, Rolerole) {

        super();

        this.userRoleId=userRoleId;

        this.user= user;

        this.role= role;

    }

    publicLonggetUserRoleId() {

        returnuserRoleId;

    }

    publicvoidsetUserRoleId(LonguserRoleId) {

        this.userRoleId=userRoleId;

    }

    publicUsergetUser() {

        return user;

    }

    publicvoidsetUser(Useruser) {

        this.user= user;

    }

    publicRolegetRole() {

        return role;

    }

    publicvoidsetRole(Rolerole) {

        this.role= role;

    }

}

1. **Repo:**

**Orderepo.java**

packagecom.medicare.repo;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.stereotype.Repository;

importcom.medicare.entities.UserOrder;

@Repository

publicinterfaceOrderRepoextendsJpaRepository<UserOrder, Long>{

    publicList<UserOrder>findByUsername(Stringusername);

}

**Productquantityrepo.java**

packagecom.medicare.repo;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.stereotype.Repository;

importcom.medicare.entities.ProductQuantity;

@Repository

publicinterfaceProductQuantityRepoextendsJpaRepository<ProductQuantity, Long>{

}

**Productrepo.java**

packagecom.medicare.repo;

importjava.util.List;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.stereotype.Repository;

importcom.medicare.entities.Product;

@Repository

publicinterfaceProductRepoextendsJpaRepository<Product, Long>{

    publicList<Product>findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(Stringname, Stringsalt);

    publicList<Product>findByCategory(Stringcategory);

    publicList<Product>findByNameAndIsAvailableTrue(Stringname);

}

**Rolerepo.java**

packagecom.medicare.repo;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.stereotype.Repository;

importcom.medicare.entities.Role;

@Repository

publicinterfaceRoleRepoextendsJpaRepository<Role, Long>{

}

**Userrepo.java**

packagecom.medicare.repo;

importorg.springframework.data.jpa.repository.JpaRepository;

importorg.springframework.stereotype.Repository;

importcom.medicare.entities.User;

@Repository

publicinterfaceUserRepoextendsJpaRepository<User, Long>{

    publicUserfindByUsername(Stringusername);

}

1. **Services:**

**Productservice.java**

packagecom.medicare.services;

importjava.util.List;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.stereotype.Service;

importcom.medicare.entities.Product;

importcom.medicare.repo.ProductRepo;

@Service

publicclassProductService {

    @Autowired

    privateProductRepoproductRepo;

    // add product

    publicProductaddProduct(Productproduct) {

        returnthis.productRepo.save(product);

    }

    //find product by id

    publicProductfindProduct(Longpid) {

        returnthis.productRepo.findById(pid).get();

    }

    //find all products

    publicList<Product>findAllProducts(){

        returnthis.productRepo.findAll();

    }

    //find product by name or salt

    publicList<Product>findByNameOrSalt(Stringname, Stringsalt){

        List<Product>products=this.productRepo.findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(name, salt);

        return products;

    }

    //find product by category

    publicList<Product>findProductByCategory(Stringcategory){

        List<Product>products=this.productRepo.findByCategory(category);

        return products;

    }

    //delete product by id

    publicvoiddeleteProductById(Longpid) {

        this.productRepo.deleteById(pid);

    }

    //find available products

    publicList<Product>findTrueProduct(Stringname){

        List<Product>products=this.productRepo.findByNameAndIsAvailableTrue(name);

        return products;

    }

}

**Userdetailservice.java**

packagecom.medicare.services;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.security.core.userdetails.UserDetails;

importorg.springframework.security.core.userdetails.UserDetailsService;

importorg.springframework.security.core.userdetails.UsernameNotFoundException;

importorg.springframework.stereotype.Service;

importcom.medicare.entities.User;

importcom.medicare.repo.UserRepo;

@Service

publicclassUserDetailServiceimplementsUserDetailsService{

    @Autowired

    privateUserRepouserRepo;

    @Override

    publicUserDetailsloadUserByUsername(Stringusername) throwsUsernameNotFoundException {

        Useruser=this.userRepo.findByUsername(username);

        if(user ==null) {

            System.out.println("User not found!");

            thrownewUsernameNotFoundException("User does not exist!");

        }

        return user;

    }

}

**Userorderservice.java**

packagecom.medicare.services;

importjava.util.List;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.stereotype.Service;

importcom.medicare.entities.ProductQuantity;

importcom.medicare.entities.UserOrder;

importcom.medicare.repo.OrderRepo;

importcom.medicare.repo.ProductQuantityRepo;

@Service

publicclassUserOrderService {

    @Autowired

    privateOrderRepoorderRepo;

    @Autowired

    privateProductQuantityRepoproductQuantityRepo;

    publicUserOrdersaveOrder(UserOrderuserOrder) {

        UserOrderorderSaved=this.orderRepo.save(userOrder);

        returnorderSaved;

    }

    publicvoidsaveProductQuantity(ProductQuantityproductQuantity) {

        this.productQuantityRepo.save(productQuantity);

    }

    publicList<UserOrder>getAll(){

        returnthis.orderRepo.findAll();

    }

    publicList<UserOrder>getUserOrders(Stringusername){

        List<UserOrder>orders=this.orderRepo.findByUsername(username);

        return orders;

    }

    publicUserOrdergetOrderById(Longoid) {

        UserOrderorder=this.orderRepo.findById(oid).get();

        return order;

    }

    publicvoiddeleteOrder(Longoid) {

        this.orderRepo.deleteById(oid);

    }

}

**Userservice.java**

packagecom.medicare.services;

importjava.util.Set;

importorg.springframework.beans.factory.annotation.Autowired;

importorg.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

importorg.springframework.stereotype.Service;

importcom.medicare.entities.User;

importcom.medicare.entities.UserRole;

importcom.medicare.repo.RoleRepo;

importcom.medicare.repo.UserRepo;

@Service

publicclassUserService {

    @Autowired

    privateUserRepouserRepo;

    @Autowired

    privateRoleReporoleRepo;

    @Autowired

    privateBCryptPasswordEncoderpasswordEncoder;

    //register a new user

    publicUsercreateUser(Useruser, Set<UserRole>userRole){

        UsernewUser=this.userRepo.findByUsername(user.getUsername());

        //if user exists or not

        try {

            if(newUser!=null) {

                thrownewException("Username already exists!");

            }else {

                //create new user

                //saving roles

                for(UserRoleuR:userRole) {

                    this.roleRepo.save(uR.getRole());

                }

                //setting userRole in user

                user.getUserRoles().addAll(userRole);

                //encoding password

                user.setPassword(this.passwordEncoder.encode(user.getPassword()));

                newUser=this.userRepo.save(user);

            }

        } catch (Exceptione) {

            System.out.println("User is already created!");

            System.out.println(e);

        }

        returnnewUser;

    }

    publicUsergetByUsername(Stringusername) {

        Useruser=this.userRepo.findByUsername(username);

        return user;

    }

}

**Resources:**

spring.datasource.name=medicare

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/medicare

spring.datasource.username=root

spring.datasource.password=Nilesh@1991

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true