MEDICARE(Source Code)

Developed By: G RajyaLAkshmi Suvarna Himavalli

AdminAdd-Product(Html):

```
<div class="container">
    <div class="row">
        <div class="col-6 mx-auto mt-3 mb-4">
            <div class="card">
                <div class="card-header">
                    <h4 class="text-center">Add a Medicine</h4>
                </div>
                <div class="card-body">
                    <form #registerationForm="ngForm">

                        <label>Medicine Continuity:</label>   
                        <div class="form-check-inline">
                            <label class="form-check-label">
                                <input type="radio" class="form-check-input" name="optradio" value="true"
                                    [(ngModel)]="product.available">Continued
                            </label>
                        </div>
                        <div class="form-check-inline">
                            <label class="form-check-label">
                                <input type="radio" class="form-check-input" name="optradio" value="false"
                                    [(ngModel)]="product.available">Discontinued
                            </label>
                        </div>
                        <hr>
                        <div class="form-group">
                            <label for="mname">Medicine Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="bname">Medicine Brand Name:</label>
                            <input type="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
```

```
<div class="form-group">
    <label for="desc">Medicine Description:</label>
    <input type="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>

<div class="form-group">
    <label for="scomp">Salt Composition:</label>
    <input type="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>

<div class="form-group">
    <label for="stk">Total Stock (In Strip):</label>
    <input type="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>

<div class="form-group">
    <label for="pc">Unit Price:</label>
    <div class="input-group">
        <div class="input-group-prepend">
            <span class="input-group-text">&#8377;</span>
        </div>
        <input type="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>
```

```
            (change)= onChangeFileField($event) >
        </div>

        <hr>
        <div class="text-center">
            <div class="btn-group">
                <button (click)="onSubmit()" class="btn btn-primary"
                    [disabled]="registerationForm.invalid" data-toggle="modal"
                    data-target="#myModal">Add Medicine</button>
                <!-- The Modal -->
                <div class="modal fade" id="myModal">
                    <div class="modal-dialog modal-dialog-centered">
                        <div class="modal-content">

                            <!-- Modal Header -->
                            <div class="modal-header">
                                <h4 class="modal-title">Add a new medicine</h4>
                                <button type="button" class="close" (click)="onClick()"
                                    data-dismiss="modal">&times;</button>
                            </div>

                            <!-- Modal body -->
                            <div class="modal-body">
                                <span *ngIf="isValid" style="color: green">{{message}}</span>
                                <span *ngIf="!isValid" style="color: red">{{message}}</span>
                            </div>

                            <!-- Modal footer -->
                            <div class="modal-footer">
                                <button type="button" class="btn btn-secondary" data-dismiss="modal"
                                    (click)="onClick()">Close</button>
                            </div>

                        </div>
                    </div>
                </div>
                <button type="reset" class="btn btn-danger">Reset</button>
            </div>
        </div>
```

Admin(AddProduct.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']
})
export class AddProductComponent {
  constructor(private userService: UserService, private router: Router) { }

  product: Product = new Product();
  file!: Blob;
  isValid!: boolean;
  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']);
  }
```

Admin-Dashboard(html):

```
<div class="row">
    <div class="col">
        <div class="card bg-light">
            <div class="card-header">
                <h3 class="text-justify">Welcome Admin: <span class="text-primary">SuvarnaValli G</span>
                         <span class="offset-4">{{username}}</span></h3>
            </div>
            <div class="card-body">
                <div class="card-deck">
                    <div class="card bg-success">
                        <img class="card-img-top" src="assets/Images/add-medicine.png" alt="add-medicine.png"
                            style="width:100%; height:250px;">
                        <div class="card-body text-center">
                            <h4 class="card-title">Add Medicine</h4>
                            <p class="card-text">Add a new medicine to our store.</p>
                        </div>
                        <div class="card-footer">
                            <a routerLink="/admin/add-medicine" role="button" class="btn btn-warning btn-block">Add
                                Medicine</a>
                        </div>
                    </div>
                    <div class="card bg-info">
                        <img class="card-img-top" src="assets/Images/all-medicines.png" alt="all-medicines.png"
                            style="width:100%; height:250px;">
                        <div class="card-body text-center">
                            <h4 class="card-title">All Medicines</h4>
                            <p class="card-text">Display all the medicine's in our store.</p>
                        </div>
                        <div class="card-footer">
                            <a routerLink="/admin/get/all/medicines" role="button"
                                class="btn btn-danger btn-block">Show All Medicines</a>
                        </div>
                    </div>
                    <div class="card bg-secondary">
                        <img class="card-img-top" src="assets/Images/order.png" alt="order.png"
                            style="width:100%; height:250px;">
                        <div class="card-body text-center">
                            <h4 class="card-title">All Orders</h4>
                            <p class="card-text">Display all the order's created by customers.</p>
                        </div>
```

Admin Dashboard.ts:

```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']
})
export class AdminDashboardComponent {
  username!: string;
  name!: string;
  constructor(private loginService: LoginService) {
    this.username = this.loginService.getUserDetails().username;
    this.name = this.loginService.getUserDetails().firstName + ' ' + this.loginService.getUserDetails().lastName;

  }

}
```

All-orders.html:

```html
<div class="col mt-2">
    <div class="table-responsive">
        <table class="table table-bordered">
            <thead class="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>
                <tr class="table-secondary" *ngFor="let o of orders">
                    <td></td>
                    <td>{{o.date}}</td>
                    <td>{{o.oid}}</td>
                    <td><br><button class="btn btn-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><button class="btn btn-danger" (click)="deleteOrder(o.oid)">Delete
                            Order</button></td>
                </tr>
            </tbody>
```

All-orders.ts:

```
@Component({
  selector: 'app-all-orders',
  templateUrl: './all-orders.component.html',
  styleUrls: ['./all-orders.component.css']
})
export class AllOrdersComponent implements OnInit {

  orders: any[] = [];
  constructor(private userService: UserService, private router: 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) {
    let url = '/order/details/' + oid;
    this.router.navigateByUrl(url);
  }

  deleteOrder(oid: number) {
    this.userService.deleteOrder(oid).subscribe({
      next: (data) => {
        this.getAllUserOrders();
      }, error: (error) => {
        console.log(error);
      }
    })
```

Show all products.html:

```
<br>
<div class="container">
    <nav class="navbar navbar-expand-sm bg-dark navbar-dark">
        <a class="navbar-brand" routerLink="/get/all/medicines" (click)="onClick()">All Medicines</a>
        <div class="dropdown dropright">
            <button type="button" class="btn btn-info dropdown-toggle" data-toggle="dropdown">
                Filter by
            </button>
            <div class="dropdown-menu">
                <a class="dropdown-item" (click)="getProductByCategory('Anti Infectives')">Anti Infectives</a>
                <a class="dropdown-item" (click)="getProductByCategory('Anti Hypertensives')">Anti Hypertensives</a>
                <a class="dropdown-item" (click)="getProductByCategory('Anti Diabetic')">Anti Diabetics</a>
                <a class="dropdown-item" (click)="getProductByCategory('Gastro Intestinal')">Gastro Intestinal</a>
                <a class="dropdown-item" (click)="getProductByCategory('Urology')">Urology</a>
                <a class="dropdown-item" (click)="getProductByCategory('Gynaecological')">Gynaecological</a>
                <a class="dropdown-item" (click)="getProductByCategory('Analgesics')">Analgesics</a>
                <a class="dropdown-item" (click)="getProductByCategory('Vitamins')">Vitamins</a>
            </div>
        </div>
        <ul class="navbar-nav ml-auto mx-4">
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbardrop" data-toggle="dropdown">
                    Sort by
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" role="button" (click)="sortByPriceLowToHigh()">Price: Low To High</a>
                    <a class="dropdown-item" role="button" (click)="sortByPriceHighToLow()">Price: High To Low</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameAscending()">Name: A-Z</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameDescending()">Name: Z-A</a>
                </div>
            </li>
        </ul>
        <form class="form-inline" (ngSubmit)="getProductByName()">
            <input class="form-control mr-sm-2" type="text" placeholder="Search" [(ngModel)]="medicineName"
                name="medicineName">
            <button class="btn btn-success" type="submit">Search</button>
        </form>
    </nav>
    <div class="table-responsive">
```

```
    </form>
</nav>
<div class="table-responsive">
    <table class="table table-light table-bordered table-hover">
        <thead class="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>
                    <img class="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>
                    <div class="btn-group">
                        <button type="button" class="btn btn-outline-success"
                            (click)="updateProduct(p.pid)">Update</button>
                        <button type="button" class="btn btn-outline-danger"
                            (click)="deleteProduct(p.pid)">Delete</button>
                    </div>
```

```
                <td>
                    <div class="btn-group">
                        <button type="button" class="btn btn-outline-success"
                            (click)="updateProduct(p.pid)">Update</button>
                        <button type="button" class="btn btn-outline-danger"
                            (click)="deleteProduct(p.pid)">Delete</button>
                    </div>
                </td>
                <td>
                    <label class="switch">
                        <input type="checkbox" name="available" [(ngModel)]="p.available"
                            (change)="onActivate(p.pid,p)">
                        <span class="slider round"></span>
                    </label>
                </td>
            </tr>
        </tbody>
    </table>
    <div class="d-flex justify-content-center">
        <pagination-controls previousLabel="Prev" nextLabel="Next" (pageChange)="onTableDataChange($event)">
        </pagination-controls>
    </div>
</div>
</div>
```

Show allproducts.ts:

```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']
})
export class ShowAllProductsComponent {

  constructor(private userService: UserService, private router: Router) {
    this.getAllProducts();
  }
  product!: Product[];
  medicineName!: string;
  prod: Product = new Product();
  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 = '' + 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 = '' + 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);
    }
  })
}
        alert('No Medicines Found');
      }
    })
  }

  deleteProduct(pid: number) {
    this.userService.deleteMedicine(pid).subscribe({
      next: (data) => {
        this.getAllProducts();
      }, error: (error) => {
        console.log(error);
      }
    })
  }
  updateProduct(pid: number) {
    let url = "/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);
      }
    })
  }
}
```

Update Product.html:

```
Go to component
<div class="container">
    <div class="row">
        <div class="col-6 mx-auto mt-3 mb-4">
            <div class="card">
                <div class="card-header">
                    <h4 class="text-center">Update Medicine</h4>
                </div>
                <div class="card-body">
                    <form #registerationForm="ngForm" (ngSubmit)="updateProduct()">

                        <div class="form-group">
                            <label for="mname">Medicine Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="bname">Medicine Brand Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="mctg">Medicine Category:</label>
                            <input type="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>

                        <div class="form-group">
                            <label for="desc">Medicine Description:</label>
                            <input type="text" class="form-control" required #desc="ngModel"
```

```
        </div>

        <div class="form-group">
            <label for="desc">Medicine Description:</label>
            <input type="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>

        <div class="form-group">
            <label for="scomp">Salt Composition:</label>
            <input type="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>

        <div class="form-group">
            <label for="stk">Total Stock (In Strip):</label>
            <input type="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>

        <div class="form-group">
            <label for="pc">Unit Price:</label>
            <div class="input-group">
                <div class="input-group-prepend">
                    <span class="input-group-text">&#8377;</span>
                </div>
                <input type="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>
        <div class="text-center">
            <button type="submit" class="btn btn-primary" [disabled]="registerationForm.invalid"
                data-toggle="modal" data-target="#myModal">Update Medicine</button>
            <!-- The Modal -->
            <div class="modal fade" id="myModal">
                <div class="modal-dialog modal-dialog-centered">
                    <div class="modal-content">

                        <!-- Modal Header -->
                        <div class="modal-header">
                            <h4 class="modal-title">Update a medicine</h4>
                            <button type="button" class="close" (click)="onClick()"
                                data-dismiss="modal">&times;</button>
                        </div>

                        <!-- Modal body -->
                        <div class="modal-body">
                            <span *ngIf="isValid" style="color: green">{{message}}</span>
                            <span *ngIf="!isValid" style="color: red">{{message}}</span>
                        </div>

                        <!-- Modal footer -->
                        <div class="modal-footer">
                            <button type="button" class="btn btn-secondary" data-dismiss="modal"
                                (click)="onClick()">Close</button>
                        </div>

                    </div>
                </div>
            </div>
        </div>
```

Update Product.ts:

```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({…
})
export class UpdateProductComponent {
  pid!: number;
  product: Product = new Product();
  isValid!: boolean;
  message!: string;
  constructor(private route: ActivatedRoute, private userService: UserService, private router: 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']);
  }
```

Adminlogin componenet.html:

```
Go to component
<div class="container">
    <div class="row">
        <div class="col-lg-4 col-md-8 mx-auto">
            <div class="card mt-5 bg-warning">
                <div class="card-body">
                    <div class="text-center">
                        <h4 class="card-title">Admin Login</h4>
                        <p>Sign in to continue.</p>
                    </div>
                    <hr>
                    <form #login="ngForm" (ngSubmit)="onSubmit()">
                        <div class="form-group">
                            <label for="email"><b>Username:</b></label> <input type="text" name="username"
                                #username="ngModel" class="form-control" placeholder="Enter Username" id="email"
                                [(ngModel)]="credentials.username" required ngModel>
                            <span *ngIf="username.invalid && username.touched" style="color:red">Please enter a valid
                                username.</span>
                        </div>
                        <div class="form-group">
                            <label for="pwd"><b>Password:</b></label> <input type="password" name="password"
                                #password="ngModel" minlength="6" class="form-control" placeholder="Enter password"
                                id="pwd" [(ngModel)]="credentials.password" required ngModel>
                            <span *ngIf="password.invalid && password.touched" style="color: red">Please enter a valid
                                password.</span>
                        </div>
                        <button type="submit" class="btn btn-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']
})
export class AdminLoginComponent {

  credentials: Credentials = new Credentials();

  constructor(private loginService: LoginService, private router: 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!');
      }
    })
  }

}
```

Cart-details.html:

```
<div class="container">
    <div class="row">
        <div class="col-lg-8 col-md-10 col-sm-12">
            <h3>Items in your cart({{totalQuantity}})</h3>
            <div *ngIf="cartItems.length == 0">
                <div class="col-6 offset-lg-6">
                    <div class="alert alert-warning text-center"><strong>Your Cart is empty!</strong></div>
                </div>
            </div>
            <div *ngIf="cartItems.length > 0">
                <table class="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>
                                <img class="img-thumbnail" width="150" height="200" [src]="item.img">
                            </td>
                            <td>
                                <h5>{{item.name}}</h5>
                                <p class="text-muted">By {{item.brand}}</p>
                                <p>MRP: <b>{{item.price | currency: 'INR'}}*</b></p>
                            </td>
                            <td><br><button class="btn btn-secondary btn-sm" (click)="increamentQuantity(item)"><i
                                        class="fas fa-plus"></i></button> <b>{{item.quantity}}</b> <button
                                    class="btn btn-secondary btn-sm" (click)="decrementQuantity(item)"><i
                                        class="fas fa-minus"></i></button>
                            </td>
                            <td><br><button class="btn btn-danger btn-sm" (click)="removeItem(item)"><i
                                        class="fas fa-trash-alt"></i></button></td>
```

```
                    </td>
                    <td><br><button class="btn btn-secondary btn-sm" (click)="increamentQuantity(item)"><i
                                class="fas fa-plus"></i></button> <b>{{item.quantity}}</b> <button
                            class="btn btn-secondary btn-sm" (click)="decrementQuantity(item)"><i
                                class="fas fa-minus"></i></button>
                    </td>
                    <td><br><button class="btn btn-danger btn-sm" (click)="removeItem(item)"><i
                                class="fas fa-trash-alt"></i></button></td>
                </tr>
            </tbody>
        </table>

    </div>

</div>
<div class="col-lg-4 col-md-12 ml-auto mt-5" *ngIf="cartItems.length > 0">
    <div class="card">
        <div class="card-header">
            <h4>Order Summary <span class="text-muted">({{totalQuantity}} item)</span></h4>
        </div>
        <div class="card-body text-justify mx-auto">
            <p class="card-text">
                Total Quantity:   {{totalQuantity}}
            </p>
            <p class="card-text">
                Delivery Charges: <span class="text-success">  Free</span>
            </p>
            <p class="card-text font-weight-bold">
                <span class="text-info">Total Payable:   {{totalPrice | currency: 'INR'}}</span>
            </p>
        </div>
        <div class="card-footer">
            <a routerLink="/user/create/order" role="button" class="btn btn-success btn-block">Proceed to Pay
                {{totalPrice | currency:
                'INR'}}</a>
        </div>
    </div>
</div>
```

Cart Details.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']
})
export class CartDetailsComponent implements OnInit {

  constructor(private cartService: 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);
  }
```

getProduct Component.html:

```
<div>
    <nav class="navbar navbar-expand-sm bg-light justify-content-center">
        <div class="col-6">
            <input type="text" class="form-control" placeholder="Search medicine, health products and more..."
                [(ngModel)]="name" (keydown.enter)="onSearch(name)">
        </div>
    </nav>
</div>

<div class="container">

    <div class="row">
        <div class="col-sm-12 col-md-10 col-lg-8 mx-auto">

            <br>
            <span style="font-size: 1.5rem;">Products</span>
            <span class="float-right mt-2"><a class="btn btn-light btn-sm dropdown-toggle" href="#"
                    data-toggle="dropdown" role="button">
                    Sort by
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" role="button" (click)="sortByPriceLowToHigh()">Price: Low To High</a>
                    <a class="dropdown-item" role="button" (click)="sortByPriceHighToLow()">Price: High To Low</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameAscending()">Name: A-Z</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameDescending()">Name: Z-A</a>
                </div>
            </span>

            <div class="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">
                <div class="card-body">
                    <h4 class="card-title">{{p.name}} <span class="float-right">&#8377; {{p.price}}</span></h4>
                    <p class="text-muted">By {{p.brand}}</p>
                    <small>Description: {{p.description}}</small>

                    <a class="dropdown-item" role="button" (click)="sortByNameDescending()">Name: Z-A</a>
                </div>
            </span>

            <div class="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">
                <div class="card-body">
                    <h4 class="card-title">{{p.name}} <span class="float-right">&#8377; {{p.price}}</span></h4>
                    <p class="text-muted">By {{p.brand}}</p>
                    <small>Description: {{p.description}}</small>
                    <br>
                    <small>Contains: <span class="font-weight-bold">{{p.salt}}</span></small>
                    <button class="btn btn-primary float-right mt-2" *ngIf="p.available" (click)="addToCart(p)">Add to
                        cart</button>
                    <button class="btn btn-secondary float-right mt-2" *ngIf="!p.available" disabled>Out of
                        Stock</button>
                </div>

            </div>
            <div class="d-flex justify-content-center mt-1">
                <pagination-controls previousLabel="Prev" nextLabel="Next" (pageChange)="onTableDataChange($event)">
                </pagination-controls>
            </div>
        </div>
    </div>
</div>
```

Getproduct.ts:

```typescript
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']
})
export class GetProductComponent implements OnInit {
  category!: string;
  product!: Product[];
  name!: string;
  page: number = 1;
  count: number = 0;
  tableSize: number = 7;
  constructor(private route: ActivatedRoute, private userService: UserService, private cartService: CartService, private router: 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 = '' + 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 = '' + p.productImage.imageData;
      })
    }, error: (error) => {
      console.log(error);
      alert('No Medicines Found');
    }
  })

 }
}

onSearch(name: string) {
  if (name != undefined) {
    console.log('navigating to search url');
    let url = "/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);
}
```

```
 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) {
   const cartItem = new CartItem(product);
   this.cartService.addToCart(cartItem);
 }

}
```

Home.html:

```
<br>
<div class="container">
    <div class="row">
        <div class="col-6 mx-auto">
            <h4>What are you looking for?</h4><br>
            <div class="input-group mb-3">
                <div class="input-group-prepend">
                    <span class="input-group-text"><img src="assets/Images/search-icon.png" alt=""></span>
                </div>
                <input type="text" class="form-control" placeholder="Search medicine, health products and more..."
                    [(ngModel)]="name" (keydown.enter)="onSearch(name)">
            </div>
        </div>
    </div>
</div>
<div id="img" class="carousel slide" data-ride="carousel">

    <ul class="carousel-indicators">
        <li data-target="#img" data-slide-to="0" class="active"></li>
        <li data-target="#img" data-slide-to="1"></li>
        <li data-target="#img" data-slide-to="2"></li>
        <li data-target="#img" data-slide-to="3"></li>
    </ul>


    <div class="carousel-inner">
        <div class="carousel-item active">
            <img src="assets/Images/pic1.jpg" alt="pic1.jpg">
        </div>
        <div class="carousel-item">
            <img src="assets/Images/pic2.jpg" alt="pic2.jpg">
        </div>
        <div class="carousel-item">
            <img src="assets/Images/pic3.jpg" alt="pic3.jpg">
        </div>
        <div class="carousel-item">
            <img src="assets/Images/pic4.jpg" alt="pic4.jpg">
        </div>
    </div>
```

```
    <a class="carousel-control-prev" href="#img" data-slide="prev">
        <span class="carousel-control-prev-icon"></span>
    </a>
    <a class="carousel-control-next" href="#img" data-slide="next">
        <span class="carousel-control-next-icon"></span>
    </a>
</div><br>
<div class="container-fluid">
    <div class="row">
        <div class="col">
            <h3 class="ml-4 pl-5">Browse medicines & health products</h3>
            <br>
            <h6 class="ml-4 pl-5">By health concerns</h6>
        </div>
    </div>
</div>
<div class="container-fluid ">
    <div class="row">
        <div class="col mb-5">
            <div id="product" class="carousel slide" data-ride="carousel" data-interval="false">
                <div class="carousel-inner">
                    <div class="carousel-item active">
                        <div class="card-deck col-11 mx-auto">
                            <div class="card bg-light">
                                <img class="card-img-top" src="assets/Images/cardiac.png" alt="cardiac.png"
                                    style="width:100%; height:130px;">
                                <div class="card-body text-center mt-1">
                                    <a routerLink="/show/product/class/{{medicine[0]}}">Cardiac Care</a>
                                </div>
                            </div>
                            <div class="card bg-light">
                                <img class="card-img-top" src="assets/Images/diabetic.png" alt="diabetic.png"
                                    style="width:100%; height:130px;">

                                <div class="card-body text-center mt-1">
                                    <a routerLink="/show/product/class/{{medicine[1]}}">Diabetic Care</a>
                                </div>
                            </div>
                            <div class="card bg-light">
                                <img class="card-img-top" src="assets/Images/gastro.png" alt="gastro.png"
                                    style="width:100%; height:130px;">

                                <div class="card-body text-center mt-1">
                                    <a routerLink="/show/product/class/{{medicine[2]}}">Gastro & Liver Care</a>
                                </div>
                            </div>

                            <div class="card bg-light">
                                <img class="card-img-top" src="assets/Images/urology.png" alt="urology.png"
                                    style="width:100%; height:130px;">

                                <div class="card-body text-center mt-1">
                                    <a routerLink="/show/product/class/{{medicine[3]}}">Urology Care</a>
                                </div>
                            </div>

                        </div>
                    </div>
                    <div class="carousel-item">
                        <div class="card-deck col-11 mx-auto">

                            <div class="card bg-light">
                                <img class="card-img-top" src="assets/Images/ai.png" alt="ai.png"
                                    style="width:100%; height:130px;">
```

```
                    <div class="card bg-light">
                        <img class="card-img-top" src="assets/Images/gynaecology.png" alt="gynaecology.png"
                            style="width:100%; height:130px;">

                        <div class="card-body text-center mt-1">
                            <a routerLink="/show/product/class/{{medicine[5]}}">Gynaecology Care</a>
                        </div>
                    </div>

                    <div class="card bg-light">
                        <img class="card-img-top" src="assets/Images/ortho.png" alt="ortho.png"
                            style="width:100%; height:130px;">

                        <div class="card-body text-center mt-1">
                            <a routerLink="/show/product/class/{{medicine[6]}}">Pain Relief</a>
                        </div>
                    </div>

                    <div class="card bg-light">
                        <img class="card-img-top" src="assets/Images/vitamins.png" alt="vitamins.png"
                            style="width:100%; height:130px;">

                        <div class="card-body text-center mt-1">
                            <a routerLink="/show/product/class/{{medicine[7]}}">Daily Health Care</a>
                        </div>
                    </div>

                </div>

            </div>

            <a class="carousel-control-prev" href="#product" data-slide="prev">
                <span class="carousel-control-prev-icon"></span>
            </a>


            <a class="carousel-control-next" href="#product" data-slide="next">
                <span class="carousel-control-next-icon"></span>
            </a>
```

```
                    </div>

                </div>

                <a class="carousel-control-prev" href="#product" data-slide="prev">
                    <span class="carousel-control-prev-icon"></span>
                </a>


                <a class="carousel-control-next" href="#product" data-slide="next">
                    <span class="carousel-control-next-icon"></span>
                </a>

            </div>
        </div>
    </div>
</div>

iv>
```

Home Component.ts:

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

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
  styleUrls: ['./home.component.css']
})
export class HomeComponent {
  constructor(public router: Router) { }
  name!: string;
  medicine: string[] = ['Anti Hypertensives', 'Anti Diabetic', 'Gastro Intestinal', 'Urology', 'Anti Infectives', 'Gynaecological', 'Analgesics'
  onSearch(name: string) {
    if (name != undefined) {
      console.log('navigating to search url');
      let url = "/user/search/product/" + name;
      this.router.navigateByUrl(url);
    } else {
      console.log('please enter a name');
    }
  }

}
```

NavBar.html:

```html
<nav class="navbar navbar-expand-lg navbar-light bg-light">

    <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarSupportedContent"
        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
        <span class="navbar-toggler-icon"></span>
    </button>

    <div class="collapse navbar-collapse" id="navbarSupportedContent">
        <a class="navbar-brand" routerLink="/"><img src="assets/Images/Medicare-Logo.png" alt="Logo"
                style="width:100px;height:45px;"></a>
        <ul class="navbar-nav mr-auto">
            <li class="nav-item active">
                <a class="nav-link" routerLink="/get/all/class/All-Medicines">Medicines<span
                        class="sr-only">(current)</span></a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Lab Tests</a>
            </li>
            <li class="nav-item dropdown">
                <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown"
                    aria-haspopup="true" aria-expanded="false">
                    Healthcare
                </a>
                <div class="dropdown-menu" aria-labelledby="navbarDropdown">
                    <a class="dropdown-item" href="#">Covid Essentials</a>
                    <a class="dropdown-item" href="#">Personal Care</a>
                    <div class="dropdown-divider"></div>
                    <a class="dropdown-item" href="#">Surgical Care</a>
                </div>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Health Blogs</a>
            </li>


        </ul>
```

```
            </li>


        </ul>
        <ul class="navbar-nav ml-auto">
            <li class="nav-item mr-1">
                <a routerLink="/get/cart/details" role="button" class="btn btn-info"><small>Cart total: {{totalPrice |
                        currency:
                        'INR'}} </small> <i class='fas fa-cart-arrow-down'></i> <span
                        class="badge badge-danger">{{totalQuantity}}</span></a>
            </li>
            <li class="nav-item mr-1">
                <a *ngIf="!loginService.isLoggedIn()" routerLink="/user/signup" role="button" class="btn btn-link">Sign
                    Up</a>
            </li>
            <li class="nav-item">
                <a *ngIf="!loginService.isLoggedIn()" routerLink="/user/login" role="button"
                    class="btn btn-outline-success">Login</a>
            </li>
            <li class="nav-item mr-1">
                <a class="btn btn-link" *ngIf="loginService.isLoggedIn()" role="button" (click)="home()"><b>Home</b></a>
            </li>
            <li class="nav-item">
                <a class="btn btn-info" *ngIf="loginService.isLoggedIn()" role="button" (click)="logout()">Logout</a>
            </li>
        </ul>
    </div>
</nav>
```

NavBar.ts:

```typescript
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { CartService } from 'src/app/Services/cart.service';
import { LoginService } from 'src/app/Services/login.service';

@Component({
  selector: 'app-navbar',
  templateUrl: './navbar.component.html',
  styleUrls: ['./navbar.component.css']
})
export class NavbarComponent implements OnInit {

  totalPrice: number = 0;
  totalQuantity: number = 0;

  constructor(public loginService: LoginService, private router: Router, private cartService: CartService) {

  }
  ngOnInit(): void {
    this.updateCartStatus();
  }

  logout() {
    this.loginService.logout();
    window.location.reload();
  }
  home() {
    if (this.loginService.getUserRole() == 'USER') {
      this.router.navigate(['/user-home']);
    } else if (this.loginService.getUserRole() == 'ADMIN') {
      this.router.navigate(['/admin-dashboard']);
    }

  }

  updateCartStatus() {
    this.cartService.totalPrice.subscribe(data => this.totalPrice = data);
    this.cartService.totalQuantity.subscribe(data => this.totalQuantity = data);
  }
```


```typescript
  logout() {
    this.loginService.logout();
    window.location.reload();
  }
  home() {
    if (this.loginService.getUserRole() == 'USER') {
      this.router.navigate(['/user-home']);
    } else if (this.loginService.getUserRole() == 'ADMIN') {
      this.router.navigate(['/admin-dashboard']);
    }

  }

  updateCartStatus() {
    this.cartService.totalPrice.subscribe(data => this.totalPrice = data);
    this.cartService.totalQuantity.subscribe(data => this.totalQuantity = data);
  }
```


Search Product.html:

```
<div class="container">
    <div class="row">
        <div class="col-sm-12 col-md-10 col-lg-8 mx-auto">

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

            <span style="font-size: 1.5rem;">Products</span>
            <span class="float-right mt-2"><a class="btn btn-light btn-sm dropdown-toggle" href="#"
                    data-toggle="dropdown" role="button">
                    Sort by
                </a>
                <div class="dropdown-menu">
                    <a class="dropdown-item" role="button" (click)="sortByPriceLowToHigh()">Price: Low To High</a>
                    <a class="dropdown-item" role="button" (click)="sortByPriceHighToLow()">Price: High To Low</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameAscending()">Name: A-Z</a>
                    <a class="dropdown-item" role="button" (click)="sortByNameDescending()">Name: Z-A</a>
                </div>
            </span>

            <div class="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">
                <div class="card-body">
                    <h4 class="card-title">{{p.name}} <span class="float-right">&#8377; {{p.price}}</span></h4>
                    <p class="text-muted">By {{p.brand}}</p>
                    <small>Description: {{p.description}}</small>
                    <br>
                    <small>Contains: <span class="font-weight-bold">{{p.salt}}</span></small>
                    <button class="btn btn-primary float-right mt-2" *ngIf="p.available" (click)="addToCart(p)">Add to
                        cart</button>
                    <button class="btn btn-secondary float-right mt-2" *ngIf="!p.available" disabled>Out of
                        Stock</button>
                </div>
```

```
                                                                                    totalItems: count
                                                                                }">
                <img [src]="p.img" class="card-img-left img-thumbnail" width="200" height="200">
                <div class="card-body">
                    <h4 class="card-title">{{p.name}} <span class="float-right">&#8377; {{p.price}}</span></h4>
                    <p class="text-muted">By {{p.brand}}</p>
                    <small>Description: {{p.description}}</small>
                    <br>
                    <small>Contains: <span class="font-weight-bold">{{p.salt}}</span></small>
                    <button class="btn btn-primary float-right mt-2" *ngIf="p.available" (click)="addToCart(p)">Add to
                        cart</button>
                    <button class="btn btn-secondary float-right mt-2" *ngIf="!p.available" disabled>Out of
                        Stock</button>
                </div>

            </div>
            <div class="d-flex justify-content-center mt-1">
                <pagination-controls previousLabel="Prev" nextLabel="Next" (pageChange)="onTableDataChange($event)">
                </pagination-controls>
            </div>
        </div>
    </div>
</div>
```

SearchProduct.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']
})
export class SearchProductComponent implements OnInit {
  medicineName!: string;
  name!: string;
  product!: Product[];
  page: number = 1;
  count: number = 0;
  tableSize: number = 7;
  constructor(private route: ActivatedRoute, private userService: UserService, private cartService: CartService, private router: 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 = '' + p.productImage.imageData;
        })
      }, error: (error) => {
        console.log(error);
        alert('No Medicines Found');
      }
      this.userService.getMedicineByName(this.medicineName).subscribe({
        next: (data) => {
          this.product = data;
          this.product.forEach((p) => {
            p.img = '' + 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) {
      const cartItem = new CartItem(product);
      this.cartService.addToCart(cartItem);
    }

}
```

Alluser-orders.html:

```
<div class="container">
    <div class="row">
        <div class="col mt-2">
            <div class="table-responsive">
                <table class="table table-bordered">
                    <thead class="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>
                        <tr class="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>
                                <button class="btn btn-success" (click)="getOrderDetails(o.oid)">View Order
                                    Details</button>
                            </td>
                        </tr>
                    </tbody>
                </table>
            </div>
        </div>
    </div>
</div>
```

alluser-order.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']
})
export class AllUserOrdersComponent implements OnInit {
  username!: string;
  orders: any[] = [];
  constructor(private route: ActivatedRoute, private userService: UserService, private router: 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) {
    let url = '/order-confirmation/invoice/' + oid;
    this.router.navigateByUrl(url);
  }
```

OrderConfirmation.html:

```
<div class="container">
    <div class="row">
        <div class="col-8 mx-auto mt-2 mb-2">
            <div class="card bg-light">
                <div class="card-header">
                    <h3 class="text-center font-weight-bold">Order Confirmation</h3>
                    <p class="text-center">Dear <b>{{orderInvoice.firstName}} {{orderInvoice.lastName}}</b>, Thank you
                        for shopping with us!</p>
                </div>
                <div class="card-body">
                    <h4 class="text-center"><b>Order Summary</b></h4>
                    <p class="text-center">{{orderInvoice.date}}</p>

                    <div class="card mx-auto">
                        <div class="card-body">
                            <p><b>Order Id: </b> {{orderInvoice.oid}} <span class="float-right"><b>Order Status:
                                    </b>{{orderInvoice.status}}</span></p>
                            <p>Ordered Products:</p>
                            <div class="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">
                                <div class="card-body">
                                    <h4 class="card-title">{{p.product.name}} <span
                                            class="float-right text-muted">&#8377;
                                            {{p.product.price * p.quantity}}</span></h4>
                                    <p class="text-muted">By {{p.product.brand}} <span
                                            class="text-dark float-right">Quantity:
                                            {{p.quantity}}</span></p>
                                </div>
                            </div>

                            <hr>
                            <div class="text-center">
                                <h5><b>Order Total: </b><span class="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>
                            <div class="text-center">
                            </div>

                            <hr>
                            <div class="text-center">
                                <h5><b>Order Total: </b><span class="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>
                            <div class="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>
                <div class="card-footer">
                    <h6 class="text-center">Your order is on it's way. Thank you!</h6>
                </div>
            </div>
        </div>
    </div>
</div>
div>
```

Order Confirmation.ts:

```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']
})
export class OrderConfirmationComponent implements OnInit {
  oid!: number;
  orderInvoice: OrderSummary = new OrderSummary();

  constructor(private route: ActivatedRoute, private userService: 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 = '' + p.product.productImage.imageData;
        })
      }, error: (error) => {
        console.log(error);
      }
    })
  }


}
```

OrderDetails.html:

```html
<div class="container">
    <div class="row">
        <div class="col-lg-6 col-md-8 mx-auto mt-3 mb-4">
            <div class="card">
                <div class="card-header">
                    <h4 class="text-center">Order Details</h4>
                </div>
                <div class="card-body">
                    <form #orderDetailForm="ngForm" (ngSubmit)="onSubmit()">

                        <div class="form-group">
                            <label for="fname">Your First Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="lname">Your Last Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="add">Your Shipping Address:</label>
                            <textarea class="form-control" rows="2" id="add" name="address"
                                [(ngModel)]="orderDetails.address" #add="ngModel"
                                placeholder="Enter your shipping address" required ngModel></textarea>
                            <span *ngIf="add.invalid && add.touched" style="color: red">Please enter a valid
                                shipping address.</span>
                        </div>

                        <div class="form-group">
                            <label for="pc">Your Area Pincode:</label>
                            <input type="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>

    <div class="form-group">
        <label for="dt">Your District:</label>
        <input type="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>

    <div class="form-group">
        <label for="ste">Your State:</label>
        <input type="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>

    <div class="form-group">
        <label for="phn">Your Contact no. :</label>
        <div class="input-group">
            <div class="input-group-prepend">
                <span class="input-group-text">+91</span>
            </div>
            <input type="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>


    <div class="text-center">
        <button type="button" class="btn btn-success btn-lg" [disabled]="orderDetailForm.invalid"
```

```
    <div class="text-center">
        <button type="button" class="btn btn-success btn-lg" [disabled]="orderDetailForm.invalid"
            data-toggle="modal" data-target="#myModal">Pay Now {{paidAmount | currency:
            'INR'}}</button>
    </div>

    <!-- The Modal -->
    <div class="modal fade" id="myModal">
        <div class="modal-dialog modal-dialog-centered">
            <div class="modal-content">

                <!-- Modal Header -->
                <div class="modal-header">
                    <h4 class="modal-title">Kindly enter your card details</h4>
                    <button type="button" class="close" data-dismiss="modal">&times;</button>
                </div>

                <!-- Modal body -->
                <div class="modal-body">
                    <div class="was-validated">
                        <div class="form-group">
                            <label for="uname">Enter the name on the card:</label>
                            <input type="text" class="form-control" id="uname"
                                placeholder="Enter the name on card" name="cname" required>
                            <div class="valid-feedback">Valid.</div>
                            <div class="invalid-feedback">Please fill out this field.
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="crd">Enter your card number:</label>
                            <input type="number" class="form-control" id="crd"
                                placeholder="Enter the card number" name="cnumber" required>
                            <div class="valid-feedback">Valid.</div>
                            <div class="invalid-feedback">Please fill out this field.
                            </div>
                        </div>
                        <div class="form-group">
                            <label for="cvvn">Enter your CVV number:</label>
```

```
                        <div class="form-group">
                            <label for="cvvn">Enter your CVV number:</label>
                            <input type="password" class="form-control" id="cvvn"
                                placeholder="Enter your cvv" name="cvv" required>
                            <div class="valid-feedback">Valid.</div>
                            <div class="invalid-feedback">Please fill out this field.
                            </div>
                        </div>

                        <div class="form-group form-check">
                            <label class="form-check-label">
                                <input class="form-check-input" type="checkbox" name="remember"
                                    required> I agree to the terms and
                                conditions.
                                <div class="valid-feedback">Valid.</div>
                                <div class="invalid-feedback">Check this checkbox to
                                    continue.</div>
                            </label>
                        </div>
                        <div>
                            <h5 class="text-center">Payable amount = {{paidAmount | currency:
                                'INR'}} </h5>
                        </div>
                        <div class="text-center">
                            <button type="submit" class="btn btn-success">Make Payment</button>
                        </div>
                    </div>
                </div>
                <!-- Modal footer -->
                <div class="modal-footer">
                    <button type="button" class="btn btn-danger"
                        data-dismiss="modal">Cancel</button>
                </div>

            </div>
        </div>
    </div>
```

Order Details.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']
})
export class OrderDetailsComponent implements OnInit {

  orderDetails: OrderDetails = new OrderDetails();
  cartItems: CartItem[] = [];
  orderItem: OrderItem[] = [];
  paidAmount: number = 0;
  username!: string;
  constructor(private cartService: CartService, private loginService: LoginService, private userService: UserService) { }
  ngOnInit(): void {
    this.cartItems = this.cartService.cartItems;
    for (let cartItems of this.cartItems) {
      let items: OrderItem = new OrderItem();
      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({
```

```
constructor(private cartService: CartService, private loginService: LoginService, private userService: UserService) { }
ngOnInit(): void {
  this.cartItems = this.cartService.cartItems;
  for (let cartItems of this.cartItems) {
    let items: OrderItem = new OrderItem();
    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);
    }
  })

}
```

UserHome.html:

```html
<div class="container">
    <div class="row">
        <div class="col">
            <div class="jumbotron jumbotron-fluid bg-secondary">
                <div class="text-center text-white">
                    <h1>Welcome {{name}} !</h1>
                </div>
            </div>
            <div class="card-deck col-9 mx-auto mb-2">
                <div class="card bg-success">
                    <img class="card-img-top" src="assets/Images/cart.png" alt="cart.png"
                        style="width:100%; height:250px;">
                    <div class="card-body text-center">
                        <h4 class="card-title">Your Cart</h4>
                        <p class="card-text">Display all the cart items.</p>
                    </div>
                    <div class="card-footer">
                        <a routerLink="/get/cart/details" role="button" class="btn btn-info btn-block">Go to Cart</a>
                    </div>
                </div>
                <div class="card bg-success">
                    <img class="card-img-top" src="assets/Images/order.png" alt="order.png"
                        style="width:100%; height:250px;">
                    <div class="card-body text-center">
                        <h4 class="card-title">All Orders</h4>
                        <p class="card-text">Display all the order's placed.</p>
                    </div>
                    <div class="card-footer">
                        <button class="btn btn-info btn-block" (click)="getOrders()">Show Orders</button>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
```

UserHome.ts:

```typescript
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']
})
export class UserHomeComponent implements OnInit {
  username!: string;
  name!: string;
  constructor(private router: Router, private loginService: LoginService) { }
  ngOnInit(): void {
    this.username = this.loginService.getUserDetails().username;
    this.name = this.loginService.getUserDetails().firstName + ' ' + this.loginService.getUserDetails().lastName;
  }

  getOrders() {
    let url = '/user/get/all-orders/' + this.username;
    this.router.navigateByUrl(url);
  }

}
```

UserLogin.html:

```
<div class="container">
    <div class="row">
        <div class="col-lg-4 col-md-8 mx-auto">

            <div class="card mt-5 bg-info">
                <div class="card-body">
                    <div class="text-center">
                        <h4 class="card-title">Welcome User</h4>
                        <p>Sign in to continue.</p>
                    </div>
                    <hr>
                    <form #login="ngForm" (ngSubmit)="onSubmit()">
                        <div class="form-group">
                            <label for="email"><b>Username:</b></label> <input type="text" name="username"
                                #username="ngModel" class="form-control" placeholder="Enter Username" id="email"
                                [(ngModel)]="credentials.username" required ngModel>
                            <span *ngIf="username.invalid && username.touched" style="color: red">Please enter a valid
                                username.</span>
                        </div>
                        <div class="form-group">
                            <label for="pwd"><b>Password:</b></label> <input type="password" name="password"
                                #password="ngModel" minlength="6" class="form-control" placeholder="Enter password"
                                id="pwd" [(ngModel)]="credentials.password" required ngModel>
                            <span *ngIf="password.invalid && password.touched" style="color: red">Please enter a valid
                                password.</span>
                        </div>
                        <button type="submit" class="btn btn-danger btn-block" [disabled]="login.invalid">Login</button>

                    </form>
                </div>
            </div>

        </div>
    </div>
</div>
```

UserLogin.ts:

src > app > Components > user-login > TS user-login.component.ts > ...

```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']
})
export class UserLoginComponent {

  credentials: Credentials = new Credentials();

  constructor(private loginService: LoginService, private router: 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!');
      }
    })
  }

}
```

UserSignup.html:

```html
<div class="container">
    <div class="row">
        <div class="col-lg-6 col-md-8 mx-auto mt-3 mb-4">
            <div class="card">
                <div class="card-header">
                    <h4 class="text-center">User Registration</h4>
                </div>
                <div class="card-body">
                    <form #registerationForm="ngForm" (ngSubmit)="onSubmit()">

                        <div class="form-group">
                            <label for="username">Your E-mail:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="pwd">Enter a Password:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="fname">First Name:</label>
                            <input type="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>
                        <div class="form-group">
                            <label for="lname">Last Name:</label>
                            <input type="text" required #lastName="ngModel" class="form-control"
```

```
            [(ngModel)]="user.firstName" ngModel>
        <span *ngIf="firstName.invalid && firstName.touched" style="color: red">Please enter valid
            first name.</span>
    </div>
    <div class="form-group">
        <label for="lname">Last Name:</label>
        <input type="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>

    <div class="form-group">
        <label for="phn">Contact no. :</label>
        <div class="input-group">
            <div class="input-group-prepend">
                <span class="input-group-text">+91</span>
            </div>
            <input type="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>

    <div class="text-center">
        <div class="btn-group">
            <button type="submit" class="btn btn-primary" [disabled]="registerationForm.invalid"
                data-toggle="modal" data-target="#myModal">Sign Up</button>
            <!-- The Modal -->
            <div class="modal fade" id="myModal">
                <div class="modal-dialog modal-dialog-centered">
                    <div class="modal-content">

                        <!-- Modal Header -->
                        <div class="modal-header">
                            <h4 class="modal-title">User Registration</h4>
                            <button type="button" class="close" (click)="onClick()"
```

```
                    <!-- The Modal -->
                    <div class="modal fade" id="myModal">
                        <div class="modal-dialog modal-dialog-centered">
                            <div class="modal-content">

                                <!-- Modal Header -->
                                <div class="modal-header">
                                    <h4 class="modal-title">User Registration</h4>
                                    <button type="button" class="close" (click)="onClick()"
                                        data-dismiss="modal">&times;</button>
                                </div>

                                <!-- Modal body -->
                                <div class="modal-body">
                                    <span *ngIf="isValid" style="color: green">{{message}}</span>
                                    <span *ngIf="!isValid" style="color: red">{{message}}</span>
                                </div>

                                <!-- Modal footer -->
                                <div class="modal-footer">
                                    <button type="button" class="btn btn-secondary" data-dismiss="modal"
                                        (click)="onClick()">Close</button>
                                </div>

                            </div>
                        </div>
                    </div>
                    <button type="reset" class="btn btn-danger">Reset</button>
                </div>
            </div>
        </form>
    </div>
</div>


</div>
</div>
iv>
```

UserSignup.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']
})
export class UserSignupComponent {

  user: User = new User();
  isValid!: boolean;
  message!: string;

  constructor(private userService: UserService, private router: 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']);
  }
}
```

AdminGuard.ts:

```ts
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';
import { LoginService } from './login.service';

@Injectable({
  providedIn: 'root'
})
export class AdminGuard implements CanActivate {

  constructor(private loginService: LoginService, private router: Router) { }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (this.loginService.isLoggedIn() && this.loginService.getUserRole() == 'ADMIN') {
      return true;
    }
    this.router.navigate(['/admin/login']);
    return false;

  }

}

```

AuthInterceptor.ts:

```ts
import { HttpEvent, HttpHandler, HttpInterceptor, HttpRequest, HTTP_INTERCEPTORS } from "@angular/common/http";
import { Injectable } from "@angular/core";
import { Observable } from "rxjs";
import { LoginService } from "./login.service";


@Injectable()
export class AuthInterceptor implements HttpInterceptor {

    constructor(private loginService: LoginService) { }

    intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {
        //add the jwt token to the request
        let authRequest = req;
        const token = this.loginService.getToken();
        if (token != null) {
            authRequest = authRequest.clone({
                setHeaders: { Authorization: `Bearer ${token}` }
            });
        }
        return next.handle(authRequest);
    }

}
export const authInterceptorProviders = [
    {
        provide: HTTP_INTERCEPTORS,
        useClass: AuthInterceptor,
        multi: true
    }
];
```

CartService.ts:

```
import { Injectable } from '@angular/core';
import { Subject } from 'rxjs';
import { CartItem } from '../cart-item';

@Injectable({
  providedIn: 'root'
})
export class CartService {

  cartItems: CartItem[] = [];
  totalPrice: Subject<number> = new Subject<number>();
  totalQuantity: Subject<number> = new Subject<number>();

  constructor() { }

  addToCart(theCartItem: CartItem) {
    //check if the cart item is already in the cart
    let alreadyInCart: boolean = false;
    let existingCartItem: CartItem | undefined = undefined;

    if (this.cartItems.length > 0) {
      //find the cart item based on the id
      existingCartItem = this.cartItems.find((item) => item.pid === theCartItem.pid);

      //check if the cart item is found
      alreadyInCart = (existingCartItem != undefined);
    }

    if (alreadyInCart) {
      //increase the quantity
      existingCartItem!.quantity++;
    } else {
      this.cartItems.push(theCartItem);
      console.log(this.cartItems);
    }
    this.calculateTotalPrice();
  }

  calculateTotalPrice() {
    let totalPriceValue: number = 0;
```

```
    }
    this.calculateTotalPrice();
  }

  calculateTotalPrice() {
    let totalPriceValue: number = 0;
    let totalQuantityValue: number = 0;

    for (let currentCartItem of this.cartItems) {
      totalPriceValue += currentCartItem.quantity * currentCartItem.price;
      totalQuantityValue += currentCartItem.quantity;
    }
    console.log(`Total price: ${totalPriceValue}, Total quantity: ${totalQuantityValue}`);
    this.totalPrice.next(totalPriceValue);
    this.totalQuantity.next(totalQuantityValue);
  }

  decrementQuantity(cartItem: CartItem) {
    cartItem.quantity--;
    if (cartItem.quantity === 0) {
      this.remove(cartItem);
    } else {
      this.calculateTotalPrice();
    }
  }

  remove(cartItem: CartItem) {
    const itemIndex = this.cartItems.findIndex(tempCartItem => tempCartItem.pid === cartItem.pid);
    if (itemIndex > -1) {
      this.cartItems.splice(itemIndex, 1);
      this.calculateTotalPrice();
    }
  }
```

LoginService.ts:

```typescript
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { Credentials } from '../credentials';

@Injectable({
  providedIn: 'root'
})
export class LoginService {

  constructor(private http: HttpClient) { }

  baseUrl = 'http://localhost:8080';

  //current user: user which is logged in
  public getCurrentUser(): Observable<any> {
    return this.http.get<any>(`${this.baseUrl}/current-user`);
  }

  //generate token
  public generateToken(credentials: Credentials): Observable<any> {
    return this.http.post<any>(`${this.baseUrl}/generate-token`, credentials);
  }

  //login user: set token in local storage
  public loginUser(token: any) {
    localStorage.setItem('token', token);
    return true;
  }

  //is logged in: user is logged in or not
  public isLoggedIn() {
    let tokenStr = localStorage.getItem('token');
    if (tokenStr == undefined || tokenStr == null || tokenStr == '') {
      return false;
    } else {
      return true;
    }
  }
```

```
  //logout user: remove token from local storage
  public logout() {
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    return true;
  }

  //get token
  public getToken() {
    return localStorage.getItem('token');
  }

  //set user details in local storage
  public setUserDetails(user: any) {
    localStorage.setItem('user', JSON.stringify(user));
  }

  //get user details from local storage
  public getUserDetails() {
    let user = localStorage.getItem('user');
    if (user != null) {
      return JSON.parse(user);
    } else {
      this.logout();
      return null;
    }
  }

  //get user role
  public getUserRole() {
    let user = this.getUserDetails();
    return user.authorities[0].authority;
  }


}
```

User.Guard.ts:

```
import { Injectable } from '@angular/core';
import { ActivatedRouteSnapshot, CanActivate, Router, RouterStateSnapshot, UrlTree } from '@angular/router';
import { Observable } from 'rxjs';
import { LoginService } from './login.service';

@Injectable({
  providedIn: 'root'
})
export class UserGuard implements CanActivate {

  constructor(private loginService: LoginService, private router: Router) { }

  canActivate(
    route: ActivatedRouteSnapshot,
    state: RouterStateSnapshot): Observable<boolean | UrlTree> | Promise<boolean | UrlTree> | boolean | UrlTree {
    if (this.loginService.isLoggedIn() && this.loginService.getUserRole() == 'USER') {
      return true;
    }
    this.router.navigate(['/user/login']);
    return false;

  }

}
```

Userservice.ts:

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
import { OrderDetails } from '../order-details';
import { Product } from '../product';
import { User } from '../user';

@Injectable({
  providedIn: 'root'
})
export class UserService {

  constructor(private http: HttpClient) { }

  baseUrl = 'http://localhost:8080';

  public userSignUp(user: User): Observable<User> {
    return this.http.post<User>(`${this.baseUrl}/user/signup`, user);
  }

  public addMedicine(product: Product, image: Blob): Observable<any> {
    let formData = new FormData();
    formData.append('product', JSON.stringify(product));
    formData.append('image', image);
    return this.http.post<any>(`${this.baseUrl}/add/product`, formData);
  }

  public getAllMedicine(): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/get/all-products`);
  }

  public getMedicineByName(name: string): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/get/products/${name}`);
  }

  public getMedicineByCategory(category: string): Observable<any[]> {
    return this.http.get<any[]>(`${this.baseUrl}/get/products-by-category/${category}`);
  }


public deleteMedicine(pid: number): Observable<any> {
  return this.http.delete<any>(`${this.baseUrl}/delete/product/${pid}`);
}

public findById(pid: number): Observable<any> {
  return this.http.get<any>(`${this.baseUrl}/get-product/${pid}`);
}

public updateMedicine(pid: number, product: Product): Observable<any> {
  return this.http.put<any>(`${this.baseUrl}/update/product/${pid}`, product);
}

public setAvailable(pid: number, product: Product): Observable<any> {
  return this.http.put<any>(`${this.baseUrl}/set-availability/product/${pid}`, product);
}

public createOrder(orderDetails: OrderDetails): Observable<OrderDetails> {
  return this.http.post<OrderDetails>(`${this.baseUrl}/user/create/order`, orderDetails);
}

public getOrderById(oid: number): Observable<any> {
  return this.http.get<any>(`${this.baseUrl}/get/order-invoice/${oid}`);
}

public getAllOrders(): Observable<any[]> {
  return this.http.get<any[]>(`${this.baseUrl}/get/all/orders`);
}

public deleteOrder(oid: number): Observable<any> {
  return this.http.delete<any>(`${this.baseUrl}/delete/order/${oid}`);
}

public getOrderByUsername(username: string): Observable<any[]> {
  return this.http.get<any[]>(`${this.baseUrl}/get/orders/${username}`);
}
```

App-routing module.ts:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AdminLoginComponent } from './Components/admin-login/admin-login.component';
import { AddProductComponent } from './Components/Admin/add-product/add-product.component';
import { AdminDashboardComponent } from './Components/Admin/admin-dashboard/admin-dashboard.component';
import { AllOrdersComponent } from './Components/Admin/all-orders/all-orders.component';
import { ShowAllProductsComponent } from './Components/Admin/show-all-products/show-all-products.component';
import { UpdateProductComponent } from './Components/Admin/update-product/update-product.component';
import { CartDetailsComponent } from './Components/cart-details/cart-details.component';
import { GetProductComponent } from './Components/get-product/get-product.component';
import { HomeComponent } from './Components/home/home.component';
import { SearchProductComponent } from './Components/search-product/search-product.component';
import { UserLoginComponent } from './Components/user-login/user-login.component';
import { UserSignupComponent } from './Components/user-signup/user-signup.component';
import { AllUserOrdersComponent } from './Components/User/all-user-orders/all-user-orders.component';
import { OrderConfirmationComponent } from './Components/User/order-confirmation/order-confirmation.component';
import { OrderDetailsComponent } from './Components/User/order-details/order-details.component';
import { UserHomeComponent } from './Components/User/user-home/user-home.component';
import { AdminGuard } from './Services/admin.guard';
import { UserGuard } from './Services/user.guard';

const routes: Routes = [
  { path: 'user/login', component: UserLoginComponent, pathMatch: 'full', title: 'User Login' },
  { path: 'admin/login', component: AdminLoginComponent, pathMatch: 'full', title: 'Admin Login' },
  { path: '', component: HomeComponent, pathMatch: 'full', title: 'Medicare' },
  { path: 'user-home', component: UserHomeComponent, canActivate: [UserGuard], pathMatch: 'full', title: 'Home' },
  { path: 'admin-dashboard', component: AdminDashboardComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'Admin Dashboard' },
  { path: 'user/signup', component: UserSignupComponent, pathMatch: 'full', title: 'User Registration' },
  { path: 'admin/add-medicine', component: AddProductComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'Add Medicine' },
  { path: 'admin/get/all/medicines', component: ShowAllProductsComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'All Medicines' },
  { path: 'admin/update/medicine/:pid', component: UpdateProductComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'Update Medicine
  { path: 'user/search/product/:name', component: SearchProductComponent, pathMatch: 'full', title: 'Search results' },
  { path: 'show/product/class/:category', component: GetProductComponent, pathMatch: 'full', title: 'All Products' },
  { path: 'get/all/class/:category', component: GetProductComponent, pathMatch: 'full', title: 'All Products' },
  { path: 'get/cart/details', component: CartDetailsComponent, pathMatch: 'full', title: 'Cart Details' },
  { path: 'user/create/order', component: OrderDetailsComponent, canActivate: [UserGuard], pathMatch: 'full', title: 'Order Details' },
  { path: 'order-confirmation/invoice/:oid', component: OrderConfirmationComponent, canActivate: [UserGuard], pathMatch: 'full', title: 'Order C
  { path: 'admin/all/user-orders', component: AllOrdersComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'All Orders' },
  { path: 'order/details/:oid', component: OrderConfirmationComponent, canActivate: [AdminGuard], pathMatch: 'full', title: 'Order Details' },
  { path: 'user/get/all-orders/:username', component: AllUserOrdersComponent, canActivate: [UserGuard], pathMatch: 'full', title: 'Orders Pl
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

AppComponent.ts:

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

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Medicare';
}
```

AppModule.ts:

```
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HttpClientModule } from '@angular/common/http';
import { NgxPaginationModule } from 'ngx-pagination';
import { authInterceptorProviders } from './Services/auth.interceptor';
import { UserLoginComponent } from './Components/user-login/user-login.component';
import { AdminLoginComponent } from './Components/admin-login/admin-login.component';
import { AdminDashboardComponent } from './Components/Admin/admin-dashboard/admin-dashboard.component';
import { UserHomeComponent } from './Components/User/user-home/user-home.component';
import { NavbarComponent } from './Components/navbar/navbar.component';
import { UserSignupComponent } from './Components/user-signup/user-signup.component';
import { AddProductComponent } from './Components/Admin/add-product/add-product.component';
import { ShowAllProductsComponent } from './Components/Admin/show-all-products/show-all-products.component';
import { UpdateProductComponent } from './Components/Admin/update-product/update-product.component';
import { HomeComponent } from './Components/home/home.component';
import { SearchProductComponent } from './Components/search-product/search-product.component';
import { GetProductComponent } from './Components/get-product/get-product.component';
import { CartDetailsComponent } from './Components/cart-details/cart-details.component';
import { OrderDetailsComponent } from './Components/User/order-details/order-details.component';
import { OrderConfirmationComponent } from './Components/User/order-confirmation/order-confirmation.component';
import { AllOrdersComponent } from './Components/Admin/all-orders/all-orders.component';
import { AllUserOrdersComponent } from './Components/User/all-user-orders/all-user-orders.component';

@NgModule({
  declarations: [
    AppComponent,
    UserLoginComponent,
    AdminLoginComponent,
    AdminDashboardComponent,
    UserHomeComponent,
    NavbarComponent,
    UserSignupComponent,
    AddProductComponent,
    ShowAllProductsComponent,
    UpdateProductComponent,
    HomeComponent,
    SearchProductComponent,
    GetProductComponent,
    CartDetailsComponent,
    OrderDetailsComponent,
    OrderConfirmationComponent,
    AllOrdersComponent,
    AllUserOrdersComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    HttpClientModule,
    NgxPaginationModule
  ],
  providers: [authInterceptorProviders],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Cartitem.ts:

```ts
import { Product } from "./product";

export class CartItem {
    pid!: number;
    name!: string;
    brand!: string;
    price!: number;
    img!: any;
    quantity!: number;

    constructor(product: Product) {
        this.pid = product.pid;
        this.name = product.name;
        this.brand = product.brand;
        this.price = product.price;
        this.img = product.img;
        this.quantity = 1;
    }
}
```

OrderDetails.ts:

```ts
import { OrderItem } from "./order-item";

export class OrderDetails {
    oid!: number;
    username!: string;
    firstName!: string;
    lastName!: string;
    address!: string;
    district!: string;
    pinCode!: number;
    state!: string;
    contact!: string;
    paidAmount!: number;
    paymentMode!: string;
    cartItem: OrderItem[] = [];

}
```

OrderItem.ts:

```ts
export class OrderItem {
    pid!: number;
    quantity!: number;
}
```

OrderSummary.ts:

```ts
import { ProductQuantity } from "./product-quantity";

export class OrderSummary {
    oid!: number;
    username!: string;
    firstName!: string;
    lastName!: string;
    address!: string;
    district!: string;
    pinCode!: number;
    state!: string;
    contact!: string;
    paidAmount!: number;
    paymentMode!: string;
    status!: string;
    date!: string;
    products: ProductQuantity[] = [];
}
```

ProductQuantity.ts:

```ts
import { Product } from "./product";

export class ProductQuantity {
    pqid!: number;
    product!: Product;
    quantity!: number;
}
```

Product.ts:

```ts
export class Product {
    pid!: number;
    available!: boolean;
    name!: string;
    brand!: string;
    category!: string;
    description!: string;
    salt!: string;
    totalAvailable!: number;
    price!: number;
    productImage: any;
    img!: any;
}
```

User.ts:

```
export class User {
    userId!: number;
    username!: string;
    password!: string;
    firstName!: string;
    lastName!: string;
    contactNumber!: string;
}
```

Index.html:

```
<base href="/">
<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="icon" type="image/x-icon" href="favicon.ico">
<link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/css/bootstrap.min.css"
  integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
/head>

body>

<app-root></app-root>


<script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"
  integrity="sha384-KJ3o2DKtIkvYIK3UENzmM7KCkRr/rE9/Qpg6aAZGJwFDMVNA/GpGFF93hXpG5KkN"
  crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.12.9/dist/umd/popper.min.js"
  integrity="sha384-ApNbgh9B+Y1QKtv3Rn7W3mgPxhU9K/ScQsAP7hUibX39j7fakFPskvXusvfa0b4Q"
  crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@4.0.0/dist/js/bootstrap.min.js"
  integrity="sha384-JZR6Spejh4U02d8jOt6vLEHfe/JQGiRRSQQxSfFWpi1MquVdAyjUar5+76PVCmYl"
  crossorigin="anonymous"></script>
  <script src="assets/all.js"></script>
<!-- <script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>-->
/body>

/html>
```

MediCare(Back-End)

MedicareBackend Application.java:

```java
package com.medicare;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class MedicareBackendApplication {

    public static void main(String[] args) {
        SpringApplication.run(MedicareBackendApplication.class, args);
    }

}
```

AuthEntryPoint.java:

```java
package com.medicare.config;

import java.io.IOException;

@Component
public class AuthEntryPoint implements AuthenticationEntryPoint{

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
            AuthenticationException authException) throws IOException, ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");

    }

}
```

ImageUtil.java:

```java
package com.medicare.config;

import java.io.IOException;

@Component
public class AuthEntryPoint implements AuthenticationEntryPoint{

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
            AuthenticationException authException) throws IOException, ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");

    }

}
```

JwtAuthFilter.java:

```java
package com.medicare.config;

import java.io.IOException;

@Component
public class AuthEntryPoint implements AuthenticationEntryPoint{

    @Override
    public void commence(HttpServletRequest request, HttpServletResponse response,
            AuthenticationException authException) throws IOException, ServletException {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED, "Unauthorized");

    }

}
```

JwtUtil.java:

```java
public class JwtUtil {

    private String SECRET_KEY = "medicare";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }
    private Claims extractAllClaims(String token) {
        return Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }

    public boolean validateToken(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return (username.equals(userDetails.getUsername()) && !isTokenExpired(token));
    }
```

SecurityConfig.java:

```java
package com.medicare.config;

import io.jsonwebtoken.Claims;

@Component
public class JwtUtil {

    private String SECRET_KEY = "medicare";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }
    private Claims extractAllClaims(String token) {
        return Jwts.parser().setSigningKey(SECRET_KEY).parseClaimsJws(token).getBody();
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());
    }

    public String generateToken(UserDetails userDetails) {
        Map<String, Object> claims = new HashMap<>();
        return createToken(claims, userDetails.getUsername());
    }

    private String createToken(Map<String, Object> claims, String subject) {

        return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 60 * 10))
                .signWith(SignatureAlgorithm.HS256, SECRET_KEY).compact();
    }
```

ValidationHandler.java:

```java
package com.medicare.config;

import java.util.HashMap;

@ControllerAdvice
public class ValidationHandler extends ResponseEntityExceptionHandler{

    @Override
    protected ResponseEntity<Object> handleMethodArgumentNotValid(MethodArgumentNotValidException ex,
            HttpHeaders headers, HttpStatus status, WebRequest request) {

        Map<String, String> errors = new HashMap<>();
        ex.getBindingResult().getAllErrors().forEach((error) ->{

            String fieldName = ((FieldError) error).getField();
            String message = error.getDefaultMessage();
            errors.put(fieldName, message);
        });
        return new ResponseEntity<Object>(errors, HttpStatus.BAD_REQUEST);
    }


}
```

JwtController.java:

```java
package com.medicare.controller;
import java.security.Principal;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.BadCredentialsException;
import org.springframework.security.authentication.DisabledException;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.medicare.config.JwtUtil;
import com.medicare.entities.JwtRequest;
import com.medicare.entities.JwtResponse;
import com.medicare.entities.User;
import com.medicare.services.UserDetailService;

@RestController
@CrossOrigin(origins = "*")
public class JwtController {

    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserDetailService userDetailService;

    @Autowired
    private JwtUtil jwtUtil;

    //generate token
    @PostMapping("/generate-token")
    public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws Exception{
        try {
            authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());
```

```java
    @Autowired
    private AuthenticationManager authenticationManager;

    @Autowired
    private UserDetailService userDetailService;

    @Autowired
    private JwtUtil jwtUtil;

    //generate token
    @PostMapping("/generate-token")
    public ResponseEntity<?> generateToken(@RequestBody JwtRequest jwtRequest) throws Exception{
        try {
            authenticate(jwtRequest.getUsername(), jwtRequest.getPassword());
        }catch(UsernameNotFoundException e) {
            e.printStackTrace();
            throw new Exception("User does not exist or invalid credentials!");
        }
        // validated
        UserDetails userDetails = this.userDetailService.loadUserByUsername(jwtRequest.getUsername());
        String token = this.jwtUtil.generateToken(userDetails);
        return ResponseEntity.ok(new JwtResponse(token));
    }
    private void authenticate(String username, String password) throws Exception {
        try {
            this.authenticationManager.authenticate(new UsernamePasswordAuthenticationToken(username, password));
        } catch (BadCredentialsException e) {
            throw new Exception("Invalid Credentials! "+e.getMessage());
        }catch(DisabledException e) {
            throw new Exception("User Disabled! "+e.getMessage());
        }
    }
    @GetMapping("/current-user")
    public User getCurrentUser(Principal principal) {
        return (User)this.userDetailService.loadUserByUsername(principal.getName());
    }
}
```

ProductController.java:

```java
package com.medicare.controller;

import java.io.IOException;

@RestController
@CrossOrigin(origins = "*")
public class ProductController {

    @Autowired
    private ProductService productService;

    @Autowired
    private ObjectMapper objectMapper;


    //add new product
    @PreAuthorize("hasAuthority('ADMIN')")
    @PostMapping("/add/product")
    public ResponseEntity<?> addNewProduct(@RequestParam("product") String product,
                                           @RequestParam("image") MultipartFile file) throws IOException{

        ProductImage img = new ProductImage();
        img.setName(file.getOriginalFilename());
        img.setType(file.getContentType());
        img.setImageData(ImageUtil.compressImage(file.getBytes()));
        Product p = null;
        try {
            p = objectMapper.readValue(product,Product.class);
            p.setProductImage(img);
        } catch (JsonMappingException e) {
            e.printStackTrace();
        } catch (JsonProcessingException e) {
            e.printStackTrace();
            return ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Invalid Request");
        }
        Product saveProduct = this.productService.addProduct(p);
        return ResponseEntity.ok(saveProduct);
    }
```

```java
    //update existing product
    @PreAuthorize("hasAuthority('ADMIN')")
    @PutMapping("/update/product/{id}")
    public ResponseEntity<?> updateProduct(@PathVariable("id") Long id,@Valid @RequestBody Product product){
        Product updateProduct = 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);
        return ResponseEntity.status(HttpStatus.CREATED).build();
    }

    //find product by id
    @GetMapping("get-product/{id}")
    public ResponseEntity<?> getProductById(@PathVariable("id") Long id){
        Product product = this.productService.findProduct(id);
        ProductImage img =  new ProductImage();
        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);
        return ResponseEntity.ok(product);
    }

    //find all products
    @GetMapping("/get/all-products")
    public ResponseEntity<?> getAllProducts(){
        List<Product> allProducts = this.productService.findAllProducts();
        allProducts.forEach(product -> {
            ProductImage img =  new ProductImage();
            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);
```

```
            img.setImgId(product.getProductImage().getImgId());
            img.setName(product.getProductImage().getName());
            img.setType(product.getProductImage().getType());
            product.setProductImage(img);
        });
        if(allProducts.isEmpty()) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }else {
            return ResponseEntity.ok(allProducts);
        }
    }

    @GetMapping(value = {"/get/products/{name}"})
    public ResponseEntity<?> getProductByName(@PathVariable("name") String name,@PathVariable("name") String salt){
        List<Product> products = this.productService.findByNameOrSalt(name, salt);
        products.forEach(product -> {
            ProductImage img =  new ProductImage();
            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()) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }else {
            return ResponseEntity.ok(products);
        }
    }
    @GetMapping("/get/products-by-category/{category}")
    public ResponseEntity<?> getProductsByCategory(@PathVariable("category") String category){
        List<Product> products = this.productService.findProductByCategory(category);
        products.forEach(product -> {
            ProductImage img =  new ProductImage();
            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);
        });
```

```
            img.setName(product.getProductImage().getName());
            img.setType(product.getProductImage().getType());
            product.setProductImage(img);
        });
        if(products.isEmpty()) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }else {
            return ResponseEntity.ok(products);
        }
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @DeleteMapping("/delete/product/{id}")
    public ResponseEntity<?> deleteProduct(@PathVariable("id") Long id){
        this.productService.deleteProductById(id);
        return ResponseEntity.status(HttpStatus.OK).build();
    }

    @PreAuthorize("hasAuthority('ADMIN')")
    @PutMapping("/set-availability/product/{id}")
    public ResponseEntity<?> setAvailability(@PathVariable("id") Long id, @RequestBody Product product){
        Product updateProduct = this.productService.findProduct(id);
        updateProduct.setAvailable(product.isAvailable());
        this.productService.addProduct(updateProduct);
        return ResponseEntity.status(HttpStatus.CREATED).build();
    }

    @GetMapping("/get/{name}")
    public ResponseEntity<?> getAvailable(@PathVariable("name") String name){
        List<Product> products = this.productService.findTrueProduct(name);
        if(products.isEmpty()) {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).build();
        }else {
            return ResponseEntity.ok(products);
        }
    }
```

UserController.java:

```java
package com.medicare.controller;

import java.net.URI;
import java.util.HashSet;
import java.util.Set;

import javax.annotation.PostConstruct;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import com.medicare.entities.Role;
import com.medicare.entities.User;
import com.medicare.entities.UserRole;
import com.medicare.services.UserService;

@RestController
@CrossOrigin(origins = "*")
public class UserController {

    @Autowired
    private UserService userService;

    //init admin user
    @PostConstruct
    public void createAdmin(){
        User admin = new User();
        admin.setUsername("admin@medicare.com");
        admin.setPassword("admin12345");
        admin.setFirstName("Suvarna");
        admin.setLastName("Valli G");
        admin.setContactNumber("6265989908");
        Role role = new Role();
```

```
        admin.setPassword("admin12345");
        admin.setFirstName("Suvarna");
        admin.setLastName("Valli G");
        admin.setContactNumber("6265989908");
        Role role = new Role();
        role.setRoleId(101L);
        role.setRoleName("ADMIN");
        UserRole ur = new UserRole();
        ur.setUser(admin);
        ur.setRole(role);
        Set<UserRole> userRole = new HashSet<>();
        userRole.add(ur);
        User adminCreated = this.userService.createUser(admin, userRole);
        System.out.println("Admin username: "+adminCreated.getUsername());
    }

    //create new user
    @PostMapping("/user/signup")
    public ResponseEntity<?> createNewUser(@Valid @RequestBody User user){
        Role role = new Role();
        role.setRoleId(102L);
        role.setRoleName("USER");
        UserRole ur = new UserRole();
        ur.setUser(user);
        ur.setRole(role);
        Set<UserRole> userRole = new HashSet<>();
        userRole.add(ur);
        if(this.userService.getByUsername(user.getUsername())!=null) {
            System.out.println("Username already exists!");
            return ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
        }else {
            User newUser = this.userService.createUser(user, userRole);
            URI location = ServletUriComponentsBuilder.fromCurrentRequest().path("/{id}").buildAndExpand(newUser.getUser
            return ResponseEntity.created(location).build();
        }
    }
```

`

UserOrderController.java:

```
package com.medicare.controller;

import java.text.DateFormat;
import java.util.Calendar;
import java.util.HashSet;
import java.util.List;
import java.util.Set;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.CrossOrigin;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RestController;

import com.medicare.config.ImageUtil;
import com.medicare.entities.CartItem;
import com.medicare.entities.CartOrder;
import com.medicare.entities.Product;
import com.medicare.entities.ProductImage;
import com.medicare.entities.ProductQuantity;
import com.medicare.entities.UserOrder;
import com.medicare.services.ProductService;
import com.medicare.services.UserOrderService;

@RestController
@CrossOrigin(origins = "*")
public class UserOrderController {

    @Autowired
    private UserOrderService userOrderService;
```

```java
@Autowired
private UserOrderService userOrderService;

@Autowired
private ProductService productService;

@PreAuthorize("hasAuthority('USER')")
@PostMapping("/user/create/order")
public ResponseEntity<?> createOrder(@Valid @RequestBody CartOrder cartOrder){

    UserOrder userOrder = new UserOrder();
    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());

    DateFormat df = DateFormat.getDateInstance();
    Calendar cl = Calendar.getInstance();
    String orderDate = 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 = new HashSet<>();
    for(CartItem item : cartItems) {
        Product product = this.productService.findProduct(item.getPid());
        int quantity = item.getQuantity();
        ProductQuantity productQuantity = new ProductQuantity();
        productQuantity.setProduct(product);
        productQuantity.setQuantity(quantity);
        this.userOrderService.saveProductQuantity(productQuantity);
        pq.add(productQuantity);
    }

    userOrder.setProducts(pq);
```

Authority.java:

```java
package com.medicare.entities;

import org.springframework.security.core.GrantedAuthority;

public class Authority implements GrantedAuthority{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    private String authority;

    public Authority(String authority) {
        super();
        this.authority = authority;
    }

    @Override
    public String getAuthority() {

        return this.authority;
    }

}
```

CartItem.java:

```java
package com.medicare.entities;

public class CartItem {

    private Long pid;
    private int quantity;
    public CartItem() {

    }
    public CartItem(Long pid, int quantity) {
        super();
        this.pid = pid;
        this.quantity = quantity;
    }
    public Long getPid() {
        return pid;
    }
    public void setPid(Long pid) {
        this.pid = pid;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }



}
```

CartOrder.java:

```java
package com.medicare.entities;

import java.util.HashSet;

public class CartOrder {

    @NotBlank
    private String username;

    @NotBlank
    private String firstName;

    @NotBlank
    private String lastName;

    @NotBlank
    private String address;

    @NotBlank
    private String district;

    @NotNull
    private int pinCode;

    @NotBlank
    private String state;

    @NotBlank
    private String contact;

    @NotNull
    private Double paidAmount;

    @NotBlank
    private String paymentMode;

    @NotEmpty
    private Set<CartItem> cartItem = new HashSet<>();

    public CartOrder() {
```

```java
    @NotEmpty
    private Set<CartItem> cartItem = new HashSet<>();

    public CartOrder() {

    }

    public CartOrder(String username, String firstName, String lastName, String address, String district, int pinCode,
            String state, String contact, Double paidAmount, String paymentMode, 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;
    }

    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
```

```
    }
    public String getDistrict() {
        return district;
    }
    public void setDistrict(String district) {
        this.district = district;
    }
    public int getPinCode() {
        return pinCode;
    }
    public void setPinCode(int pinCode) {
        this.pinCode = pinCode;
    }
    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
    public String getContact() {
        return contact;
    }
    public void setContact(String contact) {
        this.contact = contact;
    }

    public Set<CartItem> getCartItem() {
        return cartItem;
    }

    public void setCartItem(Set<CartItem> cartItem) {
        this.cartItem = cartItem;
    }

    public Double getPaidAmount() {
        return paidAmount;
    }
    public void setPaidAmount(Double paidAmount) {
        this.paidAmount = paidAmount;

    }

    public Double getPaidAmount() {
        return paidAmount;
    }
    public void setPaidAmount(Double paidAmount) {
        this.paidAmount = paidAmount;
    }
    public String getPaymentMode() {
        return paymentMode;
    }
    public void setPaymentMode(String paymentMode) {
        this.paymentMode = paymentMode;
    }


}
```

JwtRequest.java:

```java
package com.medicare.entities;

public class JwtRequest {

    String username;
    String password;

    public JwtRequest() {

    }
    public JwtRequest(String username, String password) {
        super();
        this.username = username;
        this.password = password;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getPassword() {
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }

}
```

JwtResponse.java:

```java
package com.medicare.entities;

public class JwtResponse {

    String token;

    public JwtResponse() {

    }

    public JwtResponse(String token) {
        super();
        this.token = token;
    }

    public String getToken() {
        return token;
    }

    public void setToken(String token) {
        this.token = token;
    }

}
```

Product.java:

```java
package com.medicare.entities;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Entity
@Table(name="products")
public class Product {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long pid;

    @NotBlank(message = "name cannot be blank")
    private String name;

    @NotBlank(message = "brand cannot be blank")
    private String brand;

    @NotBlank(message = "category cannot be blank")
    private String category;

    @NotBlank(message = "description cannot be blank")
    private String description;

    @NotBlank(message = "salt cannot be blank")
    private String salt;

    @NotNull(message = "available cannot be null")
    private int totalAvailable;
```

```java
@NotNull(message = "price cannot be null")
private Double price;

@NotNull(message = "isAvailable cannot be null")
private boolean isAvailable;

@OneToOne(cascade = CascadeType.ALL)
@JsonManagedReference
private ProductImage productImage;

public Product() {
    super();
}
public Product(Long pid, String name, String brand, String category,String description, String salt, int totalAvail
        boolean isAvailable, ProductImage productImage) {
    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;
}
public Long getPid() {
    return pid;
}
public void setPid(Long pid) {
    this.pid = pid;
}
public String getName() {
    return name;
}
public void setName(String name) {
    this.name = name;
}
```

```java
public String getSalt() {
    return salt;
}
public void setSalt(String salt) {
    this.salt = salt;
}
public int getTotalAvailable() {
    return totalAvailable;
}
public void setTotalAvailable(int totalAvailable) {
    this.totalAvailable = totalAvailable;
}
public Double getPrice() {
    return price;
}
public void setPrice(Double price) {
    this.price = price;
}
public boolean isAvailable() {
    return isAvailable;
}
public void setAvailable(boolean isAvailable) {
    this.isAvailable = isAvailable;
}
public ProductImage getProductImage() {
    return productImage;
}
public void setProductImage(ProductImage productImage) {
    this.productImage = productImage;
}
public String getDescription() {
    return description;
}
public void setDescription(String description) {
    this.description = description;
}
```

ProductImage.java:

```java
package com.medicare.entities;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonBackReference;

@Entity
public class ProductImage {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long imgId;

    private String name;

    private String type;

    @Lob
    @Column(name = "imagedata")
    private byte[] imageData;

    @OneToOne(mappedBy = "productImage")
    @JsonBackReference
    private Product product;

    public ProductImage() {
        super();
    }

    public ProductImage(Long imgId, String name, String type, byte[] imageData, Product product) {
        super();
        this.imgId = imgId;
        this.name = name;
        this.type = type;
        this.type = type;
        this.imageData = imageData;
        this.product = product;
    }

    public Long getImgId() {
        return imgId;
    }

    public void setImgId(Long imgId) {
        this.imgId = imgId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public byte[] getImageData() {
        return imageData;
    }

    public void setImageData(byte[] imageData) {
        this.imageData = imageData;
    }

    public Product getProduct() {
        return product;
    }
```

ProductQuantity.java:

```java
package com.medicare.entities;
import javax.persistence.Entity;

@Entity
public class ProductQuantity {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long pqid;
    @OneToOne
    private Product product;
    private int quantity;
    public ProductQuantity() {

    }
    public ProductQuantity(Long pqid, Product product, int quantity) {
        super();
        this.pqid = pqid;
        this.product = product;
        this.quantity = quantity;
    }
    public Long getPqid() {
        return pqid;
    }
    public void setPqid(Long pqid) {
        this.pqid = pqid;
    }
    public Product getProduct() {
        return product;
    }
    public void setProduct(Product product) {
        this.product = product;
    }
    public int getQuantity() {
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

}
```

Role.java:

```java
package com.medicare.entities;

import java.util.HashSet;

@Entity
@Table(name = "roles")
public class Role {
    @Id
    private Long roleId;

    @NotBlank(message = "role name cannot be null.")
    private String roleName;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.LAZY, mappedBy = "role")
    private Set<UserRole> userRoles = new HashSet<>();

    public Role() {
        super();
        // TODO Auto-generated constructor stub
    }

    public Role(Long roleId, String roleName, Set<UserRole> userRoles) {
        super();
        this.roleId = roleId;
        this.roleName = roleName;
        this.userRoles = userRoles;
    }

    public Long getRoleId() {
        return roleId;
    }

    public void setRoleId(Long roleId) {
        this.roleId = roleId;
    }

    public String getRoleName() {
        return roleName;
    }
```

```java
    public Role(Long roleId, String roleName, Set<UserRole> userRoles) {
        super();
        this.roleId = roleId;
        this.roleName = roleName;
        this.userRoles = userRoles;
    }

    public Long getRoleId() {
        return roleId;
    }

    public void setRoleId(Long roleId) {
        this.roleId = roleId;
    }

    public String getRoleName() {
        return roleName;
    }

    public void setRoleName(String roleName) {
        this.roleName = roleName;
    }

    public Set<UserRole> getUserRoles() {
        return userRoles;
    }

    public void setUserRoles(Set<UserRole> userRoles) {
        this.userRoles = userRoles;
    }


}
```

User.java:

```
package com.medicare.entities;

import java.util.Collection;

@Entity
@Table(name = "users")
public class User implements UserDetails{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userId;

    @NotBlank(message = "username cannot be null.")
    private String username;

    @NotBlank(message = "password cannot be null.")
    @Size(min = 6, message = "enter minimum six character password")
    private String password;

    @NotBlank(message = "first name cannot be null.")
    private String firstName;

    @NotBlank(message = "last name cannot be null")
    private String lastName;

    @NotBlank(message = "contact number cannot be null")
    private String contactNumber;

    private boolean enabled = true;

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy = "user")
    @JsonIgnore
    private Set<UserRole> userRoles = new HashSet<>();

    private Set<UserRole> userRoles = new HashSet<>();

    public User() {
        super();
    }

    public User(Long userId, String username, String password, String firstName, String lastName, String contactNumber,
            boolean enabled, 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;
    }

    public Long getUserId() {
        return userId;
    }

    public void setUserId(Long userId) {
        this.userId = userId;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }
```

```
    public void setPassword(String password) {
        this.password = password;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }

    public String getContactNumber() {
        return contactNumber;
    }

    public void setContactNumber(String contactNumber) {
        this.contactNumber = contactNumber;
    }

    public boolean isEnabled() {
        return enabled;
    }

    public void setEnabled(boolean enabled) {
        this.enabled = enabled;
    }

    public Set<UserRole> getUserRoles() {
        return userRoles;
    }

}

public void setUserRoles(Set<UserRole> userRoles) {
    this.userRoles = userRoles;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    Set<Authority> authority = new HashSet<>();
    this.userRoles.forEach(userRole -> {
        authority.add(new Authority(userRole.getRole().getRoleName()));
    });
    return authority;
}

@Override
public boolean isAccountNonExpired() {
    // TODO Auto-generated method stub
    return true;
}

@Override
public boolean isAccountNonLocked() {
    // TODO Auto-generated method stub
    return true;
}

@Override
public boolean isCredentialsNonExpired() {
    // TODO Auto-generated method stub
    return true;
}
```

UserOrder.java:

```java
package com.medicare.entities;

import java.util.HashSet;


@Entity
public class UserOrder {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long oid;
    private String username;
    private String firstName;
    private String lastName;
    private String address;
    private String district;
    private int pinCode;
    private String state;
    private String contact;
    private String date;
    private String status;
    private Double paidAmount;
    private String paymentMode;

    @ManyToMany(cascade = CascadeType.ALL)
    private Set<ProductQuantity> products = new HashSet<>();

    public UserOrder() {

    }

    public UserOrder(Long oid, String username, String firstName, String lastName, String address, String district,
            int pinCode, String state, String contact, String date, String status, Double paidAmount, String paymentMod
            Set<ProductQuantity> products) {
        super();
        this.oid = oid;
        this.username = username;
        this.firstName = firstName;
        this.lastName = lastName;
        this.address = address;
        this.district = district;
        this.contact = contact;
        this.date = date;
        this.status = status;
        this.paidAmount = paidAmount;
        this.paymentMode = paymentMode;
        this.products = products;
    }



    public Long getOid() {
        return oid;
    }
    public void setOid(Long oid) {
        this.oid = oid;
    }
    public String getUsername() {
        return username;
    }
    public void setUsername(String username) {
        this.username = username;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
```

```
}
public String getDistrict() {
    return district;
}
public void setDistrict(String district) {
    this.district = district;
}
public int getPinCode() {
    return pinCode;
}
public void setPinCode(int pinCode) {
    this.pinCode = pinCode;
}
public String getState() {
    return state;
}
public void setState(String state) {
    this.state = state;
}
public String getContact() {
    return contact;
}
public void setContact(String contact) {
    this.contact = contact;
}

public Set<ProductQuantity> getProducts() {
    return products;
}

public void setProducts(Set<ProductQuantity> products) {
    this.products = products;
}

public String getDate() {
    return date;
}
public void setDate(String date) {
    this.date = date;
}



public String getDate() {
    return date;
}
public void setDate(String date) {
    this.date = date;
}
public String getStatus() {
    return status;
}
public void setStatus(String status) {
    this.status = status;
}
public Double getPaidAmount() {
    return paidAmount;
}
public void setPaidAmount(Double paidAmount) {
    this.paidAmount = paidAmount;
}
public String getPaymentMode() {
    return paymentMode;
}
public void setPaymentMode(String paymentMode) {
    this.paymentMode = paymentMode;
}
```

UserRole.java:

```java
package com.medicare.entities;

import javax.persistence.Entity;

@Entity
public class UserRole {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long userRoleId;

    @ManyToOne(fetch = FetchType.EAGER)
    private User user;

    @ManyToOne
    private Role role;

    public UserRole() {
        super();

    }

    public UserRole(Long userRoleId, User user, Role role) {
        super();
        this.userRoleId = userRoleId;
        this.user = user;
        this.role = role;
    }

    public Long getUserRoleId() {
        return userRoleId;
    }

    public void setUserRoleId(Long userRoleId) {
        this.userRoleId = userRoleId;
    }

    public User getUser() {
        return user;
    }

}

public User getUser() {
    return user;
}

public void setUser(User user) {
    this.user = user;
}

public Role getRole() {
    return role;
}

public void setRole(Role role) {
    this.role = role;
}
```

OrderRepo.java:

```java
package com.medicare.repo;

import java.util.List;

@Repository
public interface OrderRepo extends JpaRepository<UserOrder, Long>{
    public List<UserOrder> findByUsername(String username);
}
```

ProductQuantityRepo.java

```java
package com.medicare.repo;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface ProductQuantityRepo extends JpaRepository<ProductQuantity, Long>{

}
```

ProductRepo.java:

```java
package com.medicare.repo;

import java.util.List;

@Repository
public interface ProductRepo extends JpaRepository<Product, Long>{
    public List<Product> findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(String name, String salt);
    public List<Product> findByCategory(String category);
    public List<Product> findByNameAndIsAvailableTrue(String name);
}
```

RoleRepo.java:

```java
package com.medicare.repo;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface RoleRepo extends JpaRepository<Role, Long>{

}
```

UserRepo.java:

```java
package com.medicare.repo;

import org.springframework.data.jpa.repository.JpaRepository;

@Repository
public interface UserRepo extends JpaRepository<User, Long>{
    public User findByUsername(String username);
}
```

ProductService.java:

```java
package com.medicare.services;

import java.util.List;

@Service
public class ProductService {

    @Autowired
    private ProductRepo productRepo;

    // add product
    public Product addProduct(Product product) {
        return this.productRepo.save(product);
    }

    //find product by id
    public Product findProduct(Long pid) {
        return this.productRepo.findById(pid).get();
    }

    //find all products
    public List<Product> findAllProducts(){
        return this.productRepo.findAll();
    }

    //find product by name or salt
    public List<Product> findByNameOrSalt(String name, String salt){
        List<Product> products = this.productRepo.findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(name, salt);
        return products;
    }

    //find product by category
    public List<Product> findProductByCategory(String category){
        List<Product> products = this.productRepo.findByCategory(category);
        return products;
    }

    //delete product by id
    public void deleteProductById(Long pid) {
        this.productRepo.deleteById(pid);
```

```java
//find product by name or salt
public List<Product> findByNameOrSalt(String name, String salt){
    List<Product> products = this.productRepo.findByNameContainingIgnoreCaseOrSaltContainingIgnoreCase(name, salt);
    return products;
}

//find product by category
public List<Product> findProductByCategory(String category){
    List<Product> products = this.productRepo.findByCategory(category);
    return products;
}

//delete product by id
public void deleteProductById(Long pid) {
    this.productRepo.deleteById(pid);
}

//find available products
public List<Product> findTrueProduct(String name){
    List<Product> products = this.productRepo.findByNameAndIsAvailableTrue(name);
    return products;
}
```

UserDetailService.java

```java
package com.medicare.services;

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

@Service
public class UserDetailService implements UserDetailsService{

    @Autowired
    private UserRepo userRepo;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        User user = this.userRepo.findByUsername(username);
        if(user == null) {
            System.out.println("User not found!");
            throw new UsernameNotFoundException("User does not exist!");
        }
        return user;
    }

}
```

UserOrderService.java:

```java
package com.medicare.services;

import java.util.List;

@Service
public class UserOrderService {

    @Autowired
    private OrderRepo orderRepo;

    @Autowired
    private ProductQuantityRepo productQuantityRepo;

    public UserOrder saveOrder(UserOrder userOrder) {
        UserOrder orderSaved = this.orderRepo.save(userOrder);
        return orderSaved;
    }
    public void saveProductQuantity(ProductQuantity productQuantity) {
        this.productQuantityRepo.save(productQuantity);
    }

    public List<UserOrder> getAll(){
        return this.orderRepo.findAll();
    }

    public List<UserOrder> getUserOrders(String username){
        List<UserOrder> orders = this.orderRepo.findByUsername(username);
        return orders;
    }

    public UserOrder getOrderById(Long oid) {
        UserOrder order = this.orderRepo.findById(oid).get();
        return order;
    }

    public void deleteOrder(Long oid) {
        this.orderRepo.deleteById(oid);
    }

}
```

UserService.java:

```java
package com.medicare.services;

import java.util.Set;

@Service
public class UserService {

    @Autowired
    private UserRepo userRepo;

    @Autowired
    private RoleRepo roleRepo;

    @Autowired
    private BCryptPasswordEncoder passwordEncoder;

    //register a new user
    public User createUser(User user, Set<UserRole> userRole){
        User newUser = this.userRepo.findByUsername(user.getUsername());
        //if user exists or not
        try {
            if(newUser!=null) {
                throw new Exception("Username already exists!");
            }else {
                //create new user

                //saving roles
                for(UserRole uR : 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 (Exception e) {

            newUser = this.userRepo.save(user);

        }
    } catch (Exception e) {
        System.out.println("User is already created!");
        System.out.println(e);
    }

    return newUser;
}

public User getByUsername(String username) {
    User user = this.userRepo.findByUsername(username);
    return user;
}
```

Application.properties:

```
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=Shivavalli@1
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
```