**Frontend-Angular**

1. **Component: -**

**navbar.component.html**

<header>

    <navclass=”navbar navbar-expand-sm bg-dark navbar-dark”>

        <a\*ngIf=”!loggedIn”class=”navbar-brand”routerLink=”/”>Kitchen Story</a>

        <a\*ngIf=”loggedIn”class=”navbar-brand”routerLink=”/dashboard”>Admin Dashboard</a>

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

            <liclass=”nav-item”>

                <a\*ngIf=”!loggedIn”routerLink=”/login”role=”button”class=”btn btn-danger”>Admin Login</a>

                <a\*ngIf=”loggedIn”(click)=”logoutAdmin()”role=”button”class=”btn btn-danger”>Logout</a>

            </li>

        </ul>

    </nav>

</header>

**Navbar.component.ts**

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

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

@Component({

  selector:‘app-navbar’,

  templateUrl:‘./navbar.component.html’,

  styleUrls: [‘./navbar.component.css’]

})

exportclassNavbarComponentimplementsOnInit {

  publicloggedIn = false;

  constructor(privateloginService: LoginService) {

  }

  ngOnInit(): void {

    this.loggedIn = this.loginService.isLoggedIn()

  }

  logoutAdmin() {

    this.loginService.logout()

    location.reload()

  }

}

**login.component.html**

<divclass=”container”>

    <divclass=”row”>

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

            <divclass=”card mt-1 bg-warning”>

                <imgclass=”card-img-top”src=”https://www.w3schools.com/bootstrap4/img\_avatar1.png”alt=”Card image”

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

                <divclass=”card-body”>

                    <h4class=”card-title text-center”>Kitchen Story Admin Login</h4>

                    <hr>

                    <form(ngSubmit)=”onSubmit()”>

                        <divclass=”form-group”>

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

                                [(ngModel)]=”credentials.username”class=”form-control”placeholder=”Enter Username”

                                id=”email”required>

                        </div>

                        <divclass=”form-group”>

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

                                [(ngModel)]=”credentials.password”class=”form-control”placeholder=”Enter password”

                                id=”pwd”required>

                        </div>

                        <buttontype=”submit”class=”btn btn-primary btn-block”>Login</button>

                        <arouterLink=”/change-password”role=”button”class=”btn btn-outline-danger btn-block”>Change

                            Password</a>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**Login.component.ts**

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

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

@Component({

  selector:‘app-login’,

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

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

})

exportclassLoginComponent {

  credentials = {

    username:‘’,

    password:‘’

  }

  constructor(privateloginService: LoginService) { }

  onSubmit() {

    if ((this.credentials.username != ‘’&&this.credentials.password != ‘’) && (this.credentials.username != null&&this.credentials.password != null)) {

      //token generate

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

        (response: any) => {

          //success

          console.log(response.token);

          this.loginService.loginUser(response.token)

          window.location.href = “/dashboard”

        }, error=> {

          //error

          console.log(error);

          alert(“Invalid credentials!”)

        }

      )

    } else {

      alert(“Kindly fill details”);

    }

  }

}

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

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col”>

            <divclass=”jumbotron text-center bg-info”>

                <h1class=”display-3”><b>Kitchen Story</b></h1>

                <p>We’re here to serve only the best products for you.</p>

            </div>

        </div>

    </div>

    <br>

    <divclass=”col-8 mx-auto”>

        <divclass=”card bg-warning”>

            <divclass=”card-header”>

                <h5class=”text-center”>Search essentials, grocery and more...</h5>

            </div>

            <divclass=”card-body”>

                <divclass=”form-group col-8 mx-auto my-auto”>

                    <inputtype=”text”class=”form-control”[(ngModel)]=”product”name=”product”

                        placeholder=”Search for products...”>

                </div>

            </div>

            <divclass=”card-footer text-center”><button(click)=”search(product)”

                    class=”btn btn-success btn-lg”>Search</button></div>

        </div>

    </div>

</div>

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

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

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

@Component({

  selector:‘app-home’,

  templateUrl:‘./home.component.html’,

  styleUrls: [‘./home.component.css’]

})

exportclassHomeComponent {

  product: any;

  constructor(privaterouter: Router) { }

  search(name: string) {

    if (name == null) {

      alert(‘Please enter a product name!’);

    } else {

      leturl: string = “/product-search/” + name;

      this.router.navigateByUrl(url);

    }

  }

}

**dashboard.component.html**

<br><br><br><br>

<main>

    <divclass=”container”>

        <divclass=”row”>

            <divclass=”col”>

                <divclass=”card mx-auto bg-light”>

                    <divclass=”card-body”>

                        <h1class=”card-title”>Admin Dashboard</h1>

                        <pclass=”card-text”>Welcome <b>Admin</b>.Love to see you again!</p>

                        <divclass=”card-deck”>

                            <divclass=”card bg-info”>

                                <divclass=”card-body”>

                                    <h4class=”card-title”>Add Grocery Items</h4>

                                    <pclass=”card-text”>Add new items to your stock list. </p>

                                    <arouterLink=”/add-item”role=”button”class=”btn btn-danger btn-block”><b>Add

                                            Items</b>

                                    </a>

                                </div>

                            </div>

                            <divclass=”card bg-warning”>

                                <divclass=”card-body”>

                                    <h4class=”card-title”>Show Grocery Items</h4>

                                    <pclass=”card-text”>Display all available grocery items.</p>

                                    <arouterLink=”/all-items”role=”button”class=”btn btn-danger btn-block”><b>Show

                                            Items</b></a>

                                </div>

                            </div>

                            <divclass=”card bg-success”>

                                <divclass=”card-body”>

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

                                    <pclass=”card-text”>Display all orders created by users.</p>

                                    <arouterLink=”/all-orders”role=”button”class=”btn btn-danger btn-block”><b>Show

                                            Orders</b></a>

                                </div>

                            </div>

                        </div>

                    </div>

                </div>

            </div>

        </div>

    </div>

</main>

**Dashboard.component.ts**

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

@Component({

  selector:‘app-dashboard’,

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

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

})

exportclassDashboardComponent {

}

**changepassword.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-lg-5 col-md-6 mx-auto mt-5”>

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

                <divclass=”card-body”>

                    <h4class=”card-title text-center”>Admin Change Password</h4>

                    <hr>

                    <form(ngSubmit)=”onSubmit()”>

                        <divclass=”form-group”>

                            <labelfor=”email”>Username:</label><inputtype=”text”name=”username”class=”form-control”

                                [(ngModel)]=”admin.username”placeholder=”Enter Username”required>

                        </div>

                        <divclass=”form-group”>

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

                                [(ngModel)]=”admin.password”class=”form-control”placeholder=”Enter new password”

                                id=”pwd1”required>

                        </div>

                        <divclass=”form-group”>

                            <labelfor=”pwd”>Confirm New Password:</label><inputtype=”password”name=”cnfPassword”

                                [(ngModel)]=”cnfPassword”class=”form-control”placeholder=”Confirm new password”

                                id=”pwd2”required>

                        </div>

                        <buttontype=”submit”class=”btn btn-danger btn-block”>Change Password</button>

                        <arouterLink=”/login”role=”button”class=”btn btn-outline-success btn-block”>Back

                            to Admin Login</a>

                    </form>

                </div>

            </div>

        </div>

    </div>

</div>

**changepassword.component.ts**

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

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

import { Admin } from‘./admin’;

@Component({

  selector:‘app-changepassword’,

  templateUrl:‘./changepassword.component.html’,

  styleUrls: [‘./changepassword.component.css’]

})

exportclassChangepasswordComponent {

  admin: Admin = newAdmin();

  cnfPassword: any;

  constructor(privateloginService: LoginService) { }

  changePassword() {

    this.loginService.changePassword(this.admin).subscribe({

      next: (response) => {

        console.log(response);

        alert(‘Password changed successfully!’);

      }, error: (error) => {

        console.log(error);

        alert(‘Username is invalid!’);

      }

    })

  }

  onSubmit() {

    if (this.admin.username == null || this.admin.password == null) {

      alert(“Please fill in your username and new password”)

    } elseif (this.admin.password != this.cnfPassword) {

      alert(“Password does not match!”);

    } else {

      this.changePassword()

    }

  }

}

**additem.component.html**

<main>

    <divclass=”container”>

        <divclass=”row”>

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

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

                    <divclass=”card-body”>

                        <h3class=”card-title text-center”>Insert Grocery Item Details:</h3>

                        <hr>

                        <form(ngSubmit)=”onSubmit()”>

                            <divclass=”form-row”>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”pid”>Product Id</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.pid”name=”pid”placeholder=”Enter product id”required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”pname”>Product name</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.name”name=”name”placeholder=”Enter product name”required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”type”>Category</label><select[(ngModel)]=”item.category”

                                        name=”category”class=”form-control”>

                                        <optionselected>Choose..</option>

                                        <optionvalue=”Fruits & Vegetables”>Fruits & Vegetables</option>

                                        <optionvalue=”Dairy & Bakery”>Dairy & Bakery</option>

                                        <optionvalue=”Staples”>Staples</option>

                                        <optionvalue=”Snacks”>Snacks</option>

                                        <optionvalue=”Beverages”>Beverages</option>

                                        <optionvalue=”Personal Care”>Personal Care</option>

                                    </select>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”ing”>Ingredient</label><select[(ngModel)]=”item.ingredient”

                                        name=”ingredient”class=”form-control”>

                                        <optionselected>Choose..</option>

                                        <optionvalue=”Vegetarian”>Vegetarian</option>

                                        <optionvalue=”Non-Vegetarian”>Non-Vegetarian</option>

                                    </select>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”price”>Price(in Rs.)</label><inputtype=”number”class=”form-control”

                                        [(ngModel)]=”item.price”name=”price”placeholder=”Enter product M.R.P. in Rs.”

                                        Required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”manft”>Manufacturer</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.brand”name=”brand”placeholder=”Enter product manufacturer”

                                        required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”quant”>Product Quantity</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.quantity”name=”quantity”placeholder=”Enter product quantity”

                                        required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”origin”>Country Of Origin</label><inputtype=”text”

                                        [(ngModel)]=”item.origin”class=”form-control”name=”origin”

                                        placeholder=”Enter product country of origin”required>

                                </div>

                            </div>

                            <divclass=”text-center”>

                                <divclass=”btn-group”role=”group”aria-label=”Basic example”>

                                    <buttontype=”submit”class=”btn btn-primary btn-lg”>Add Item</button>

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

                                </div>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

</main>

**additem.component.ts**

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

import { AdminService } from‘src/app/Services/admin.service’;

import { Item } from‘../allitems/Items’;

@Component({

  selector:‘app-additem’,

  templateUrl:‘./additem.component.html’,

  styleUrls: [‘./additem.component.css’]

})

exportclassAdditemComponent {

  item: Item = newItem();

  constructor(privateadminService: AdminService) { }

  saveItem() {

    this.adminService.addItem(this.item).subscribe({

      next: (item) => {

        console.log(“Added item”);

      }, error: (errors) => {

        console.log(errors);

      }

    })

  }

  onSubmit() {

    if (this.item.name == null || this.item.brand == null || this.item.origin == null || this.item.price == null || this.item.quantity == null || this.item.pid == null || this.item.category == null || this.item.ingredient == null) {

      alert(“Please Enter Complete Details!”)

    } else {

      console.log(this.item);

      this.saveItem()

      alert(“Item Added Successfully!”)

    }

  }

}

**Update-item.component.html**

<main>

    <divclass=”container”>

        <divclass=”row”>

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

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

                    <divclass=”card-body”>

                        <h3class=”card-title text-center”>Update Grocery Item Details:</h3>

                        <hr>

                        <form(ngSubmit)=”onSubmit()”>

                            <divclass=”form-row”>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”pid”>Product Id</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.pid”name=”pid”placeholder=”Enter product id”required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”pname”>Product name</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.name”name=”name”placeholder=”Enter product name”required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”type”>Category</label><select[(ngModel)]=”item.category”

                                        name=”category”class=”form-control”>

                                        <optionselected>Choose..</option>

                                        <optionvalue=”Fruits & Vegetables”>Fruits & Vegetables</option>

                                        <optionvalue=”Dairy & Bakery”>Dairy & Bakery</option>

                                        <optionvalue=”Staples”>Staples</option>

                                        <optionvalue=”Snacks”>Snacks</option>

                                        <optionvalue=”Beverages”>Beverages</option>

                                        <optionvalue=”Personal Care”>Personal Care</option>

                                    </select>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”ing”>Ingredient</label><select[(ngModel)]=”item.ingredient”

                                        name=”ingredient”class=”form-control”>

                                        <optionselected>Choose..</option>

                                        <optionvalue=”Vegetarian”>Vegetarian</option>

                                        <optionvalue=”Non-Vegetarian”>Non-Vegetarian</option>

                                    </select>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”price”>Price(in Rs.)</label><inputtype=”number”class=”form-control”

                                        [(ngModel)]=”item.price”name=”price”placeholder=”Enter product M.R.P. in Rs.”

                                        Required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”manft”>Manufacturer</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.brand”name=”brand”placeholder=”Enter product manufacturer”

                                        required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”quant”>Product Quantity</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”item.quantity”name=”quantity”placeholder=”Enter product quantity”

                                        required>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”origin”>Country Of Origin</label><inputtype=”text”

                                        [(ngModel)]=”item.origin”class=”form-control”name=”origin”

                                        placeholder=”Enter product country of origin”required>

                                </div>

                            </div>

                            <divclass=”text-center”>

                                <divclass=”btn-group”role=”group”aria-label=”Basic example”>

                                    <buttontype=”submit”class=”btn btn-primary btn-lg”>Update Item</button>

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

                                </div>

                            </div>

                        </form>

                    </div>

                </div>

            </div>

        </div>

    </div>

</main>

**Update-item.component.ts**

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

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

import { Item } from‘../allitems/Items’;

import { AdminService } from‘src/app/Services/admin.service’;

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

@Component({

  selector:‘app-update-item’,

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

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

})

exportclassUpdateItemComponentimplementsOnInit {

  name!: string;

  item: Item = newItem();

  constructor(privateadminService: AdminService, privateroute: ActivatedRoute) { }

  ngOnInit(): void {

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

    this.adminService.getItemByName(this.name).subscribe({

      next: (data) => {

        this.item = data;

      }, error: (error) => {

        console.log(error);

      }

    })

  }

  onSubmit() {

    this.adminService.updateItem(this.name, this.item).subscribe({

      next: (data) => {

        console.log(data);

        alert(“Item updated successfully!”);

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**Product-search.component.html**

<divclass=”container”>

    <divclass=”row mx-auto mt-4”>

        <divclass=”col”>

            <divclass=”card-columns”>

                <div\*ngFor=”let product of item”class=”card bg-warning”>

                    <imgclass=” card-img-top”

                        src=”https://img.freepik.com/free-vector/shopping-supermarket-cart-with-grocery-pictogram\_1284-11697.jpg?w=740&t=st=1672168343~exp=1672168943~hmac=4018e3def7957bf35104fa91ff39940cb701fbe95e5483966941a83644f52fc0”

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

                    <divclass=”card-header”>

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

                    </div>

                    <divclass=”card-body”>

                        <pclass=”card-text”><b>Quantity: </b>{{product.quantity}}</p>

                        <pclass=”card-text”><b>Price:</b><span> Rs </span> {{product.price}}</p>

                        <pclass=”card-text”><b>Ingeredient: </b>{{product.ingredient}}</p>

                        <pclass=”card-text”><b>Manufacturer: </b>{{product.brand}}</p>

                        <pclass=”card-text”><b>Country of origin: </b>{{product.origin}}</p>

                    </div>

                    <divclass=”card-footer”>

                        <buttonclass=”btn btn-primary btn-block”(click)=”selectItem(product.name)”><b>Buy

                                Now</b></button>

                    </div>

                </div>

            </div>

        </div>

    </div>

</div>

**Product-search.component.ts**

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

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

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

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

import { Item } from‘../allitems/Items’;

@Component({

  selector:‘app-product-search’,

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

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

})

exportclassProductSearchComponentimplementsOnInit {

  name!: string;

  item!: Item[];

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

  ngOnInit(): void {

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

    this.showItems()

  }

  showItems() {

    this.userService.searchItems(this.name).subscribe({

      next: (items: Item[]) => {

        this.item = items;

        console.log(this.item);

      }, error: (error: any) => {

        console.log(error);

        alert(‘No products found!’);

      }

    })

  }

  selectItem(item: string) {

    leturl = “/place-order/” + item;

    this.router.navigateByUrl(url);

  }

}

**Allitems.component.html**

<br>

<main>

    <divclass=”container”>

        <divclass=”row”>

            <divclass=”col”>

                <h2class=”text-center text-white bg-dark”>All Grocery Items</h2>

                <divclass=”table-responsive”>

                    <tableclass=”table table-dark table-striped table-bordered table-hover”>

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

                            <tr>

                                <th>S.No.</th>

                                <th>Product Id</th>

                                <th>Product Name</th>

                                <th>Category</th>

                                <th>Ingredient</th>

                                <th>Price(Rs.)</th>

                                <th>Manufacturer</th>

                                <th>Quantity</th>

                                <th>Country Origin</th>

                                <th>Action</th>

                            </tr>

                        </thead>

                        <tbodyclass=”text-center”>

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

                                <td></td>

                                <td><span>{{item.pid}}</span></td>

                                <td><span>{{item.name}}</span></td>

                                <td><span>{{item.category}}</span></td>

                                <td><span>{{item.ingredient}}</span></td>

                                <td><span>{{item.price}}</span></td>

                                <td><span>{{item.brand}}</span></td>

                                <td><span>{{item.quantity}}</span></td>

                      <td><span>{{item.origin}}</span></td>

                                <td>

                                    <divclass=”btn-group”role=”group”aria-label=”Basic example”>

                                        <button(click)=”deleteItem(item.name)”class=”btn btn-danger”>Delete</button>

                                        <button(click)=”updateItem(item.name)”class=”btn btn-info”>Update</button>

                                    </div>

                                </td>

                            </tr>

                        </tbody>

                    </table>

                </div>

            </div>

        </div>

    </div>

</main>

**Allitems.component.ts**

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

import { AdminService } from‘src/app/Services/admin.service’;

import { Item } from‘./Items’;

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

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

@Component({

  selector:‘app-allitems’,

  templateUrl:‘./allitems.component.html’,

  styleUrls: [‘./allitems.component.css’]

})

exportclassAllitemsComponentimplementsOnInit {

  allItems!: Item[];

  constructor(privateadminService: AdminService, privaterouter: Router) {

  }

  ngOnInit(): void {

    this.getItemsList()

  }

  getItemsList() {

    this.adminService.getItems().subscribe({

      next: (items) => {

        this.allItems = items;

      }, error: (errors) => {

        console.log(errors);

        alert(“No items found!”);

      }

    })

  }

  updateItem(name: string) {

    leturl: string = “/update-item/” + name;

    this.router.navigateByUrl(url);

  }

  deleteItem(name: string) {

    this.adminService.deleteItem(name).subscribe({

      next: (data) => {

        this.getItemsList();

      }, error: (error) => { console.log(error) }

    })

  }

}

**Items.ts**

exportclassItem {

    name!: string;

    pid!: string;

    category!: string;

    ingredient!: string;

    price!: number;

    brand!: string;

    quantity!: string;

    origin!: string;

}

**All-orders.component.html**

<br>

<main>

    <divclass=”container”>

        <divclass=”row”>

            <divclass=”col”>

                <h2class=”text-center text-white bg-dark”>All Orders</h2>

                <divclass=”table-responsive”>

                    <tableclass=”table table-dark table-striped table-bordered table-hover”>

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

                            <tr>

                                <th>Order Id</th>

                                <th>Date</th>

                                <th>Customer Name</th>

                                <th>Address</th>

                                <th>State</th>

                                <th>Contact</th>

                                <th>Quantity</th>

                                <th>Payment</th>

                                <th>Product(Id)</th>

                                <th>Weight</th>

                                <th>Action</th>

                            </tr>

                        </thead>

                        <tbodyclass=”text-center”>

                            <tr\*ngFor=”let order of orders”>

                                <td><span>{{order.id}}</span></td>

                                <td><span>{{order.od}}</span></td>

                                <td><span>{{order.name}}</span></td>

                                <td><span>{{order.address}}</span></td>

                                <td><span>{{order.state}}</span></td>

                                <td><span>{{order.contact}}</span></td>

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

                                <td><span>{{order.payment}}</span></td>

                                <td><span>{{order.product.name}} ({{order.product.pid}})</span></td>

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

                                <td>

                                    <button(click)=”deleteOrder(order.id)”class=”btn btn-danger”>Delete</button>

                                </td>

                            </tr>

                        </tbody>

                    </table>

                </div>

            </div>

        </div>

    </div>

</main>

**All-order.component.ts**

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

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

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

import { Order } from‘../place-order/order’;

@Component({

  selector:‘app-all-orders’,

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

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

})

exportclassAllOrdersComponentimplementsOnInit {

  orders!: Order[];

  constructor(privateuserService: UserService) { }

  ngOnInit(): void {

    this.getAllOrders()

  }

  getAllOrders() {

    this.userService.getAllOrders().subscribe({

      next: (data) => {

        this.orders = data

      }, error: (error) => {

        alert(‘No orders found!’)

        console.log(error);

      }

    })

  }

  deleteOrder(id: string) {

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

      next: (data) => {

        console.log(‘Successfully deleted’);

        this.getAllOrders();

      }, error: (error) => {

        console.log(error);

      }

    })

  }

}

**Place-order.component.html**

<main>

    <divclass=”container”>

        <divclass=”row”>

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

                <divclass=”card mt-2 bg-light”>

                    <divclass=”card-header”>

                        <h5class=”text-center”><b>Product:

                            </b>{{product.name}}<label>&nbsp;&nbsp;&nbsp;

                                <span><b>Quantity: </b>{{product.quantity}}</span></label>

                        </h5>

                    </div>

                    <divclass=”card-body”>

                        <h3class=”card-title text-center”>Please enter your details to create order:</h3>

                        <hr>

                        <form(ngSubmit)=”onSubmit()”class=”was-validated”>

                            <divclass=”form-row”>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”name”>Enter your full name:</label><inputtype=”text”

                                        class=”form-control”[(ngModel)]=”order.name”name=”name”

                                        placeholder=”Enter your full name”required>

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

                                    <divclass=”invalid-feedback”>Please fill out this field.</div>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”address”>Enter shipping address:</label><inputtype=”text”

                                        class=”form-control”[(ngModel)]=”order.address”name=”address”

                                        placeholder=”Enter your address”required>

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

                                    <divclass=”invalid-feedback”>Please fill out this field.</div>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”st”>Enter State:</label><inputtype=”text”class=”form-control”

                                        [(ngModel)]=”order.state”name=”state”placeholder=”Enter your state”required>

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

                                    <divclass=”invalid-feedback”>Please fill out this field.</div>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”cnt”>Enter Contact:</label><inputtype=”number”class=”form-control”

                                        [(ngModel)]=”order.contact”name=”contact”placeholder=”Enter your contact”

                                        required>

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

                                    <divclass=”invalid-feedback”>Please fill out this field.</div>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”qty”>Select quantity</label><select[(ngModel)]=”order.quantity”

                                        name=”quantity”class=”form-control”>

                                        <optionvalue=”1”>1</option>

                                        <optionvalue=”2”>2</option>

                                        <optionvalue=”3”>3</option>

                                        <optionvalue=”4”>4</option>

                                        <optionvalue=”5”>5</option>

                                    </select>

                                </div>

                                <divclass=”form-group col-md-6”>

                                    <labelfor=”pmt”>Choose payment mode:</label><select[(ngModel)]=”order.payment”

                                        (click)=”onPay()”name=”payment”class=”form-control”>

                                        <optionselected>Choose..</option>

                                        <optionvalue=”COD”>Cash On Delivery</option>

                                        <optionvalue=”Card”>Card Payment</option>

                                    </select>

                                </div>

                            </div>

                            <div>

                                <divclass=”text-center”><button\*ngIf=”payment”type=”submit”

                                        class=”btn btn-success btn-lg”>Place order</button>

                                    <button\*ngIf=”!payment”type=”button”class=”btn btn-success btn-lg”

                                        data-toggle=”modal”data-target=”#myModal”>

                                        Pay now

                                    </button>

                                </div>

                                <!—The Modal 🡪

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

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

                                        <divclass=”modal-content”>

                                            <!—Modal Header 🡪

                                            <divclass=”modal-header”>

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

                                                <buttontype=”button”class=”close”

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

                                            </div>

                                            <!—Modal body 🡪

                                            <divclass=”modal-body”>

                                                <form(ngSubmit)=”onSubmit()”class=”was-validated”>

                                                    <divclass=”form-group”>

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

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

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

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

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

                                                        </div>

                                                    </div>

                                                    <divclass=”form-group”>

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

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

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

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

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

                                                        </div>

                                                    </div>

                                                    <divclass=”form-group”>

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

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

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

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

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

                                                        </div>

                                                    </div>

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

                                                        <labelclass=”form-check-label”>

                                                            <inputclass=”form-check-input”type=”checkbox”

                                                                name=”remember”required> I agree to the terms and

                                                            conditions.

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

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

                                                                continue.</div>

                                                        </label>

                                                    </div>

                                                    <div>

                                                        <h5>Payable amount =

                                                            Rs. {{(this.order.quantity) \* (this.product.price)}}

                                                        </h5>

                                                    </div>

                                                    <buttontype=”submit”class=”btn btn-success”>Make Payment</button>

                                                </form>

                                            </div>

                                            <!—Modal footer 🡪

                                            <divclass=”modal-footer”>

                                                <buttontype=”button”class=”btn btn-danger”

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

                                            </div>

                                        </div>

                                    </div>

                                </div>

                            </div>

                        </form>

                    </div>

                    <divclass=”card-footer”>

                        <h5class=”text-center”>Total payable amount = Rs. {{(this.order.quantity) \*

                            (this.product.price)}}

                        </h5>

                    </div>

                </div>

            </div>

        </div>

    </div>

</main>

**Place-order.component.ts**

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

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

import { AdminService } from‘src/app/Services/admin.service’;

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

import { Item } from‘../allitems/Items’;

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

import { Order } from‘./order’;

@Component({

  selector:‘app-place-order’,

  templateUrl:‘./place-order.component.html’,

  styleUrls: [‘./place-order.component.css’]

})

exportclassPlaceOrderComponentimplementsOnInit {

  item!: string;

  product: Item = newItem();

  order: Order = newOrder();

  amount!: number;

  publicpayment = false;

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

  ngOnInit(): void {

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

    this.adminService.getItemByName(this.item).subscribe({

      next: (item) => {

        this.product = item;

        console.log(this.product);

      }, error: (error) => {

        console.log(error);

      }

    })

  }

  onPay(): oolean {

    if (this.order.payment == “COD”) {

      this.payment = true;

      returnthis.payment;

    } else {

      returnthis.payment = false;

    }

  }

  onSubmit(): void {

    if (this.order.name == null || this.order.address == null || this.order.contact == null || this.order.state == null) {

      alert(“Please enter the details!”);

    } else {

      this.order.id = newDate().getTime().toString();

      this.order.product = this.product;

      this.userService.orderProduct(this.order).subscribe({

        next: (product) => {

          leturl = “/order-confirm/” + this.order.id;

          this.router.navigateByUrl(url).then(() => { location.reload() });

        }, error: (error) => {

          console.log(error);

        }

      })

    }

  }

}

**Oder.ts**

import { Item } from“../allitems/Items”;

exportclassOrder {

    id!: string;

    name!: string;

    address!: string;

    state!: string;

    contact!: string;

    quantity!: number;

    payment!: string;

    od!: string;

    product!: Item;

}

**Order-confirm.component.html**

<divclass=”container”>

    <divclass=”row”>

        <divclass=”col-lg-8 mx-auto mt-3”>

            <divclass=”card  bg-light”>

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

                    <h3><b>Order Confirmation</b></h3>

                    <p>Dear <b>{{getOrder.name}}</b>, Thank you for shopping with us!</p>

                </div>

                <divclass=”card-body”>

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

                    <pclass=”text-center”>{{getOrder.od}}</p>

                    <divclass=”card mx-auto”>

                        <divclass=”card-body”>

                            <pclass=”text-center”><b>Order Id: </b> {{getOrder.id}} <span>&nbsp;</span><b>Product:

                                </b> {{product.name}}

                            </p>

                            <pclass=”text-center”><b>Weight: </b> {{product.quantity}} <span>&nbsp;</span><b>Price:

                                </b>Rs.

                                {{product.price}}

                                <span>&nbsp;</span><b>Ordered quantity: </b> {{getOrder.quantity}}

                            </p>

                            <hr>

                            <divclass=”text-center”>

                                <h5><b>Order Total</b></h5>

                                <p><b>Payment mode: </b> {{getOrder.payment}}</p>

                                <p><b>Total Price: </b>Rs. {{(getOrder.quantity)\*(product.price)}}</p>

                            </div>

                            <hr>

                            <divclass=”text-center”>

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

                                <p><b>Shipping address: </b>{{getOrder.address}},{{getOrder.state}}</p>

                                <p><b>Contact detail: </b>{{getOrder.contact}}</p>

                            </div>

                        </div>

                    </div>

                </div>

                <divclass=”card-footer”>

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

                </div>

            </div>

        </div>

    </div>

</div>

**Order-confirm.component.ts**

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

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

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

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

import { Order } from‘../place-order/order’;

import { Item } from‘../allitems/Items’;

@Component({

  selector:‘app-order-confirm’,

  templateUrl:‘./order-confirm.component.html’,

  styleUrls: [‘./order-confirm.component.css’]

})

exportclassOrderConfirmComponentimplementsOnInit {

  id!: number;

  getOrder: Order = newOrder();

  product: Item = newItem();

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

  ngOnInit(): void {

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

    this.userService.getOrder(this.id).subscribe({

      next: (order) => {

        this.getOrder = order

        this.product = this.getOrder.product

        console.log(this.getOrder)

      }, error: (error) => {

        console.log(error)

      }

    })

  }

}

1. **Services: -**

**Admin.service.ts**

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

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

import { Observable } from'rxjs';

import { Item } from'../Components/allitems/Items';

@Injectable({

  providedIn:'root'

})

exportclassAdminService {

  baseUrl='http://localhost:9595'

  constructor(privatehttp:HttpClient) { }

  getItems():Observable<Item[]> {

    returnthis.http.get<Item[]>(`${this.baseUrl}/getItems`)

  }

  addItem(item:Item):Observable<Object> {

    returnthis.http.post<Object>(`${this.baseUrl}/saveItem`, item)

  }

  getItemByName(name:string):Observable<Item> {

    returnthis.http.get<Item>(`${this.baseUrl}/getItem/${name}`)

  }

  updateItem(name:string, item:Item):Observable<Object> {

    returnthis.http.put<Object>(`${this.baseUrl}/updateItem/${name}`, item)

  }

  deleteItem(name:string):Observable<Object> {

    returnthis.http.delete<Object>(`${this.baseUrl}/deleteItem/${name}`)

  }

}

**Auth.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'

})

exportclassAuthGuardimplementsCanActivate {

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

  canActivate(

    route:ActivatedRouteSnapshot,

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

    if (this.loginService.isLoggedIn()) {

      returntrue;

    }

    this.router.navigate(['login'])

    returnfalse;

  }

}

**Login.service.ts**

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

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

import { Observable } from'rxjs';

import { Admin } from'../Components/changepassword/admin';

@Injectable({

  providedIn:'root'

})

exportclassLoginService {

  url="http://localhost:9595"

  constructor(privatehttp:HttpClient) { }

  generateToken(credentials:any) {

    returnthis.http.post(`${this.url}/token`, credentials)

  }

  loginUser(token:any) {

    localStorage.setItem('token', token);

    returntrue;

  }

  isLoggedIn() {

    lettoken=localStorage.getItem('token');

    if (token==undefined||token===''||token==null) {

      returnfalse;

    } else {

      returntrue;

    }

  }

  logout() {

    localStorage.removeItem('token');

    returntrue;

  }

  getToken() {

    returnlocalStorage.getItem('token');

  }

  changePassword(admin:Admin):Observable<Object> {

    returnthis.http.put<Object>(`${this.url}/changePassword`, admin);

  }

}

**user.service.ts**

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

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

import { Observable } from'rxjs';

import { Item } from'../Components/allitems/Items';

import { Order } from'../Components/place-order/order';

@Injectable({

  providedIn:'root'

})

exportclassUserService {

  baseUrl='http://localhost:9595'

  constructor(privatehttp:HttpClient) { }

  searchItems(name:string):Observable<Item[]> {

    returnthis.http.get<Item[]>(`${this.baseUrl}/getProduct/${name}`)

  }

  orderProduct(order:Order):Observable<Object> {

    returnthis.http.post<Object>(`${this.baseUrl}/order`, order)

  }

  getOrder(id:number):Observable<Order> {

    returnthis.http.get<Order>(`${this.baseUrl}/getOrder/${id}`)

  }

  getAllOrders():Observable<Order[]> {

    returnthis.http.get<Order[]>(`${this.baseUrl}/getOrders`)

  }

  deleteOrder(id:string):Observable<Object> {

    returnthis.http.delete<Object>(`${this.baseUrl}/delete/${id}`)

  }

}

**Backend-SpringBoot**

**Config:**

**CustomerUserDetails.java**

packagecom.kitchenstory.config;

importjava.util.Collection;

importjava.util.HashSet;

importorg.springframework.security.core.GrantedAuthority;

importorg.springframework.security.core.authority.SimpleGrantedAuthority;

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

importcom.kitchenstory.entities.AdminCred;

@SuppressWarnings(“serial”)

publicclassCustomUserDetailsimplementsUserDetails{

    privateAdminCredadminCred;

    publicCustomUserDetails(AdminCredadminCred) {

        this.adminCred= adminCred;

    }

    @Override

    publicCollection<?ExtendsGrantedAuthority>getAuthorities() {

        HashSet<SimpleGrantedAuthority>set=newHashSet<>();

        set.add(newSimpleGrantedAuthority(this.adminCred.getRole()));

        return set;

    }

    @Override

    publicStringgetPassword() {

        // TODO Auto-generated method stub

        returnthis.adminCred.getPassword();

    }

    @Override

    publicStringgetUsername() {

        // TODO Auto-generated method stub

        returnthis.adminCred.getUsername();

    }

    @Override

    publicooleanisAccountNonExpired() {

        // TODO Auto-generated method stub

        returntrue;

    }

    @Override

    publicooleanisAccountNonLocked() {

        // TODO Auto-generated method stub

        returntrue;

    }

    @Override

    publicooleanisCredentialsNonExpired() {

        // TODO Auto-generated method stub

        returntrue;

    }

    @Override

    publicooleanisEnabled() {

        // TODO Auto-generated method stub

        returntrue;

    }

}

**jwtAuthEntryPoint.java**

packagecom.kitchenstory.config;

importjava.io.IOException;

importjavax.servlet.ServletException;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

importorg.springframework.security.core.AuthenticationException;

importorg.springframework.security.web.AuthenticationEntryPoint;

importorg.springframework.stereotype.Component;

@Component

publicclassJwtAuthEntryPointimplementsAuthenticationEntryPoint{

    @Override

    publicvoidcommence(HttpServletRequestrequest, HttpServletResponseresponse,

            AuthenticationExceptionauthException) throwsIOException, ServletException {

        response.sendError(401,”Unauthorized”);

    }

}

Jwtaurhfilter.java

packagecom.kitchenstory.config;

importjava.io.IOException;

importjavax.servlet.FilterChain;

importjavax.servlet.ServletException;

importjavax.servlet.http.HttpServletRequest;

importjavax.servlet.http.HttpServletResponse;

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

importorg.springframework.security.authentication.UsernamePasswordAuthenticationToken;

importorg.springframework.security.core.context.SecurityContextHolder;

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

importorg.springframework.security.web.authentication.WebAuthenticationDetailsSource;

importorg.springframework.stereotype.Component;

importorg.springframework.web.filter.OncePerRequestFilter;

importcom.kitchenstory.jwthelper.JwtUtil;

importcom.kitchenstory.services.CustomUserDetailService;

@Component

publicclassJwtAuthFilterextendsOncePerRequestFilter {

    @Autowired

    CustomUserDetailServicecustomUserDetailService;

    @Autowired

    privateJwtUtiljwtUtil;

    @Override

    protectedvoiddoFilterInternal(HttpServletRequestrequest, HttpServletResponseresponse, FilterChainfilterChain)

            throwsServletException, IOException {

        StringrequestTokenHeader=request.getHeader("Authorization");

        Stringusername=null;

        StringjwtToken=null;

        if(requestTokenHeader!=null&&requestTokenHeader.startsWith("Bearer ")) {

            jwtToken =requestTokenHeader.substring(7);

            try {

                username =this.jwtUtil.extractUsername(jwtToken);

            }catch(Exceptione) {

                e.printStackTrace();

            }

            UserDetailsuserDetails=this.customUserDetailService.loadUserByUsername(username);

            if(username!=null&&SecurityContextHolder.getContext().getAuthentication()==null) {

                UsernamePasswordAuthenticationTokenusernamePasswordAuthenticationToken=newUsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());

                usernamePasswordAuthenticationToken.setDetails(newWebAuthenticationDetailsSource().buildDetails(request));

                SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);

            }else {

                System.out.println("Token is invalid!");

            }

        }

        filterChain.doFilter(request, response);

    }

}

**Securityconfig.java**

packagecom.kitchenstory.config;

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

importorg.springframework.context.annotation.Bean;

importorg.springframework.context.annotation.Configuration;

importorg.springframework.security.authentication.AuthenticationManager;

importorg.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

importorg.springframework.security.config.annotation.web.builders.HttpSecurity;

importorg.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

importorg.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

importorg.springframework.security.config.http.SessionCreationPolicy;

importorg.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

importorg.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

importcom.kitchenstory.services.CustomUserDetailService;

@SuppressWarnings("deprecation")

@Configuration

@EnableWebSecurity

publicclassSecurityConfigextendsWebSecurityConfigurerAdapter{

    @Autowired

    privateCustomUserDetailServicecustomUserDetailService;

    @Autowired

    privateJwtAuthEntryPointentryPoint;

    @Autowired

    JwtAuthFilterauthFilter;

    @Override

    protectedvoidconfigure(HttpSecurityhttp) throwsException {

        http

            .csrf()

            .disable()

            .cors().and()

            .authorizeRequests()

            .antMatchers("/token").permitAll()

            .antMatchers("/changePassword").permitAll()

            .antMatchers("/getProduct/\*\*").permitAll()

            .antMatchers("/getItem/\*\*").permitAll()

            .antMatchers("/order").permitAll()

            .antMatchers("/getOrder/\*\*").permitAll()

            .anyRequest().authenticated()

            .and().sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)

            .and().exceptionHandling().authenticationEntryPoint(entryPoint);

        http.addFilterBefore(authFilter, UsernamePasswordAuthenticationFilter.class);

    }

    @Override

    protectedvoidconfigure(AuthenticationManagerBuilderauth) throwsException {

        auth.userDetailsService(this.customUserDetailService).passwordEncoder(passwordEncoder());

    }

    @Bean

    publicBCryptPasswordEncoderpasswordEncoder() {

        returnnewBCryptPasswordEncoder(10);

    }

    @Bean

    publicAuthenticationManagerauthenticationManagerBean() throwsException {

        returnsuper.authenticationManagerBean();

    }

}

Controller:

packagecom.kitchenstory.controller;

importjava.util.List;

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

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

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

importcom.kitchenstory.entities.Product;

importcom.kitchenstory.services.ProductService;

@RestController

@CrossOrigin(origins ="\*")

publicclassAdminLoginController {

    @Autowired

    privateProductServiceprodService;

    @PostMapping("/saveItem")

    publicResponseEntity<?>saveItem(@RequestBodyProductproduct){

        this.prodService.addProduct(product);

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

    }

    @GetMapping("/getItems")

    publicResponseEntity<?>getItems() {

        List<Product>allItems=this.prodService.showProducts();

        if(allItems.isEmpty()) {

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

        }else {

            returnResponseEntity.ok(allItems);

        }

    }

    @DeleteMapping("/deleteItem/{name}")

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

        try {

            this.prodService.deleteProduct(name);

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

        }catch(Exceptione) {

            System.out.println("Empty Result Data Access Exception.");

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

        }

    }

    @PutMapping("/updateItem/{name}")

    publicResponseEntity<?>updateItem(@PathVariable("name") Stringname,@RequestBodyProductproduct) {

        try {

            this.prodService.updateProduct(name, product);

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

        }catch(Exceptione) {

            System.out.println("Empty Result Data Access Exception.");

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

        }

    }

    @GetMapping("/getItem/{name}")

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

        Productproduct=this.prodService.getProduct(name);

        if(product!=null) {

            returnResponseEntity.ok(product);

        }else {

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

        }

    }

}

Jwtcontroller.java

packagecom.kitchenstory.controller;

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

importorg.springframework.http.ResponseEntity;

importorg.springframework.security.authentication.AuthenticationManager;

importorg.springframework.security.authentication.UsernamePasswordAuthenticationToken;

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

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

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

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

importorg.springframework.web.bind.annotation.RequestMapping;

importorg.springframework.web.bind.annotation.RequestMethod;

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

importcom.kitchenstory.entities.JwtRequest;

importcom.kitchenstory.entities.JwtResponse;

importcom.kitchenstory.jwthelper.JwtUtil;

importcom.kitchenstory.services.CustomUserDetailService;

@RestController

@CrossOrigin(origins ="\*")

publicclassJwtController {

    @Autowired

    AuthenticationManagerauthenticationManager;

    @Autowired

    CustomUserDetailServicecustomUserDetailService;

    @Autowired

    JwtUtiljwtUtil;

    @RequestMapping(value="/token",method =RequestMethod.POST)

    publicResponseEntity<?>generateToken(@RequestBodyJwtRequestjwtRequest){

        System.out.println(jwtRequest);

        try {

            this.authenticationManager.authenticate(newUsernamePasswordAuthenticationToken(jwtRequest.getUsername(), jwtRequest.getPassword()));

        }catch(UsernameNotFoundExceptione) {

            e.printStackTrace();

        }

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

        Stringtoken=this.jwtUtil.generateToken(userDetails);

        System.out.println("Token: "+token);

        returnResponseEntity.ok(newJwtResponse(token));

    }

}

Singleincontroller.java

packagecom.kitchenstory.controller;

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

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

importorg.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

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

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

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

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

importcom.kitchenstory.entities.AdminCred;

importcom.kitchenstory.services.AdminService;

@RestController

@CrossOrigin(origins ="\*")

publicclassSignInController {

    @Autowired

    privateAdminServiceadminService;

    @Autowired

    privateBCryptPasswordEncoderpasswordEncoder;

    @PutMapping("/changePassword")

    publicResponseEntity<?>setPassword(@RequestBodyAdminCredadmin) {

        AdminCredadm=this.adminService.findAdmin(admin.getUsername());

        if(adm ==null) {

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

        }

        else {

            adm.setPassword(this.passwordEncoder.encode(admin.getPassword()));

            this.adminService.saveCred(adm);

            returnResponseEntity.ok(adm);

        }

    }

}

Usercontroller.java

packagecom.kitchenstory.controller;

importjava.text.DateFormat;

importjava.util.Calendar;

importjava.util.List;

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

importorg.springframework.http.HttpStatus;

importorg.springframework.http.ResponseEntity;

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

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

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

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

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

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

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

importcom.kitchenstory.entities.Order;

importcom.kitchenstory.entities.Product;

importcom.kitchenstory.services.OrderService;

importcom.kitchenstory.services.ProductService;

@RestController

@CrossOrigin(origins ="\*")

publicclassUserController {

    @Autowired

    privateOrderServiceservice;

    @Autowired

    privateProductServicepservice;

    @PostMapping("/order")

    publicResponseEntity<?>createOrder(@RequestBodyOrderorder){

        DateFormatdf=DateFormat.getDateInstance();

        Calendarcl=Calendar.getInstance();

        Stringod=df.format(cl.getTime());

        order.setOd(od);

        this.service.saveOrder(order);

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

    }

    @GetMapping("/getOrders")

    publicResponseEntity<?>getAllOrders(){

        List<Order>orders=this.service.getOrders();

        if(orders.isEmpty()) {

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

        }

        else {

            returnResponseEntity.ok(orders);

        }

    }

    @DeleteMapping("/delete/{id}")

    publicResponseEntity<?>deleteOrder(@PathVariable("id") Stringid){

        this.service.deleteOrder(id);

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

    }

    @GetMapping("/getProduct/{name}")

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

        List<Product>products=this.pservice.getAllByName(name);

        if(products.isEmpty()) {

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

        }else {

            returnResponseEntity.ok(products);

        }

    }

    @GetMapping("/getOrder/{id}")

    publicResponseEntity<?>getOrder(@PathVariable("id") Stringid){

        Orderorder=this.service.getOrder(id);

        if(order==null) {

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

        }else {

            returnResponseEntity.ok(order);

        }

    }

}

Entites:

Admincred.java

packagecom.kitchenstory.entities;

importorg.springframework.data.annotation.Id;

importorg.springframework.data.mongodb.core.mapping.Document;

@Document(collection ="AdminCred")

publicclassAdminCred {

    @Id

    privateStringusername;

    privateStringpassword;

    privateStringrole;

    publicAdminCred() {

        super();

        // TODO Auto-generated constructor stub

    }

    publicAdminCred(Stringusername, Stringpassword, Stringrole) {

        super();

        this.username= username;

        this.password= password;

        this.role= role;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetPassword() {

        return password;

    }

    publicvoidsetPassword(Stringpassword) {

        this.password= password;

    }

    publicStringgetRole() {

        return role;

    }

    publicvoidsetRole(Stringrole) {

        this.role= role;

    }

}

Jwtrequest.java

packagecom.kitchenstory.entities;

publicclassJwtRequest {

    Stringusername;

    Stringpassword;

    publicJwtRequest() {

    }

    publicJwtRequest(Stringusername, Stringpassword) {

        super();

        this.username= username;

        this.password= password;

    }

    publicStringgetUsername() {

        return username;

    }

    publicvoidsetUsername(Stringusername) {

        this.username= username;

    }

    publicStringgetPassword() {

        return password;

    }

    publicvoidsetPassword(Stringpassword) {

        this.password= password;

    }

    @Override

    publicStringtoString() {

        return"JwtRequest [username="+ username +", password="+ password +"]";

    }

}

Jwtresponse.java

packagecom.kitchenstory.entities;

publicclassJwtResponse {

    Stringtoken;

    publicJwtResponse() {

        super();

    }

    publicJwtResponse(Stringtoken) {

        super();

        this.token= token;

    }

    publicStringgetToken() {

        return token;

    }

    publicvoidsetToken(Stringtoken) {

        this.token= token;

    }

}

Order.java

packagecom.kitchenstory.entities;

importorg.springframework.data.annotation.Id;

importorg.springframework.data.mongodb.core.mapping.DBRef;

importorg.springframework.data.mongodb.core.mapping.Document;

@Document(collection ="Orders")

publicclassOrder {

    @Id

    privateStringid;

    privateStringname;

    privateStringaddress;

    privateStringstate;

    privateStringcontact;

    privateStringod;

    privateintquantity;

    privateStringpayment;

    @DBRef

    privateProductproduct;

    publicOrder() {

        super();

        // TODO Auto-generated constructor stub

    }

    publicOrder(Stringid, Stringname, Stringaddress, Stringstate, Stringcontact, Stringod, intquantity,

            Stringpayment, Productproduct) {

        super();

        this.id= id;

        this.name= name;

        this.address= address;

        this.state= state;

        this.contact= contact;

        this.od= od;

        this.quantity= quantity;

        this.payment= payment;

        this.product= product;

    }

    publicStringgetId() {

        return id;

    }

    publicvoidsetId(Stringid) {

        this.id= id;

    }

    publicStringgetName() {

        return name;

    }

    publicvoidsetName(Stringname) {

        this.name= name;

    }

    publicStringgetAddress() {

        return address;

    }

    publicvoidsetAddress(Stringaddress) {

        this.address= address;

    }

    publicStringgetState() {

        return state;

    }

    publicvoidsetState(Stringstate) {

        this.state= state;

    }

    publicStringgetContact() {

        return contact;

    }

    publicvoidsetContact(Stringcontact) {

        this.contact= contact;

    }

    publicStringgetOd() {

        return od;

    }

    publicvoidsetOd(Stringod) {

        this.od= od;

    }

    publicintgetQuantity() {

        return quantity;

    }

    publicvoidsetQuantity(intquantity) {

        this.quantity= quantity;

    }

    publicStringgetPayment() {

        return payment;

    }

    publicvoidsetPayment(Stringpayment) {

        this.payment= payment;

    }

    publicProductgetProduct() {

        return product;

    }

    publicvoidsetProduct(Productproduct) {

        this.product= product;

    }

}

Product.java

packagecom.kitchenstory.entities;

importorg.springframework.data.annotation.Id;

importorg.springframework.data.mongodb.core.mapping.Document;

@Document(collection ="Products")

publicclassProduct {

    @Id

    privateStringname;

    privateStringpid;

    privateStringcategory;

    privateStringingredient;

    privateintprice;

    privateStringbrand;

    privateStringquantity;

    privateStringorigin;

    publicProduct() {

    }

    publicProduct(Stringname, Stringpid, Stringcategory, Stringingredient, intprice, Stringbrand,

            Stringquantity, Stringorigin) {

        super();

        this.name= name;

        this.pid= pid;

        this.category= category;

        this.ingredient= ingredient;

        this.price= price;

        this.brand= brand;

        this.quantity= quantity;

        this.origin= origin;

    }

    publicStringgetName() {

        return name;

    }

    publicvoidsetName(Stringname) {

        this.name= name;

    }

    publicStringgetPid() {

        return pid;

    }

    publicvoidsetPid(Stringpid) {

        this.pid= pid;

    }

    publicStringgetCategory() {

        return category;

    }

    publicvoidsetCategory(Stringcategory) {

        this.category= category;

    }

    publicStringgetIngredient() {

        return ingredient;

    }

    publicvoidsetIngredient(Stringingredient) {

        this.ingredient= ingredient;

    }

    publicintgetPrice() {

        return price;

    }

    publicvoidsetPrice(intprice) {

        this.price= price;

    }

    publicStringgetBrand() {

        return brand;

    }

    publicvoidsetBrand(Stringbrand) {

        this.brand= brand;

    }

    publicStringgetQuantity() {

        return quantity;

    }

    publicvoidsetQuantity(Stringquantity) {

        this.quantity= quantity;

    }

    publicStringgetOrigin() {

        return origin;

    }

    publicvoidsetOrigin(Stringorigin) {

        this.origin= origin;

    }

}

Jwtutil.java

packagecom.kitchenstory.jwthelper;

importio.jsonwebtoken.Claims;

importio.jsonwebtoken.Jwts;

importio.jsonwebtoken.SignatureAlgorithm;

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

importorg.springframework.stereotype.Component;

importjava.util.Date;

importjava.util.HashMap;

importjava.util.Map;

importjava.util.function.Function;

@Component

publicclassJwtUtil {

     privateStringSECRET\_KEY="secret";

        publicStringextractUsername(Stringtoken) {

            returnextractClaim(token, Claims::getSubject);

        }

        publicDateextractExpiration(Stringtoken) {

            returnextractClaim(token, Claims::getExpiration);

        }

        public<T>TextractClaim(Stringtoken, Function<Claims, T>claimsResolver) {

            finalClaimsclaims=extractAllClaims(token);

            returnclaimsResolver.apply(claims);

        }

        privateClaimsextractAllClaims(Stringtoken) {

            returnJwts.parser().setSigningKey(SECRET\_KEY).parseClaimsJws(token).getBody();

        }

        privateBooleanisTokenExpired(Stringtoken) {

            returnextractExpiration(token).before(newDate());

        }

        publicStringgenerateToken(UserDetailsuserDetails) {

            Map<String, Object>claims=newHashMap<>();

            returncreateToken(claims, userDetails.getUsername());

        }

        privateStringcreateToken(Map<String, Object>claims, Stringsubject) {

            returnJwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(newDate(System.currentTimeMillis()))

                    .setExpiration(newDate(System.currentTimeMillis() +1000\*60\*60\*10))

                    .signWith(SignatureAlgorithm.HS256, SECRET\_KEY).compact();

        }

        publicBooleanvalidateToken(Stringtoken, UserDetailsuserDetails) {

            finalStringusername=extractUsername(token);

            return (username.equals(userDetails.getUsername()) &&!isTokenExpired(token));

        }

}

Repo:

Admincredrepo.java

packagecom.kitchenstory.repo;

importorg.springframework.data.mongodb.repository.MongoRepository;

importorg.springframework.stereotype.Repository;

importcom.kitchenstory.entities.AdminCred;

@Repository

publicinterfaceAdminCredRepoextendsMongoRepository<AdminCred, String> {

    publicAdminCredfindByUsername(Stringusername);

}

Orderrepo.java

packagecom.kitchenstory.repo;

importorg.springframework.data.mongodb.repository.MongoRepository;

importorg.springframework.stereotype.Repository;

importcom.kitchenstory.entities.Order;

@Repository

publicinterfaceOrderRepoextendsMongoRepository<Order,String>{

}

Productrepo.java

packagecom.kitchenstory.repo;

importorg.springframework.data.mongodb.repository.MongoRepository;

importorg.springframework.stereotype.Repository;

importcom.kitchenstory.entities.Product;

@Repository

publicinterfaceProductRepoextendsMongoRepository<Product, String>{

    publicProductfindByName(Stringname);

    publicIterable<Product>findByNameContainingIgnoreCase(Stringname);

}

Services

packagecom.kitchenstory;

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

importorg.springframework.boot.CommandLineRunner;

importorg.springframework.boot.SpringApplication;

importorg.springframework.boot.autoconfigure.SpringBootApplication;

importorg.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

importcom.kitchenstory.entities.AdminCred;

importcom.kitchenstory.repo.AdminCredRepo;

@SpringBootApplication

publicclassPhase4KitchenStoryApplicationimplementsCommandLineRunner {

    @Autowired

    privateAdminCredRepocredRepo;

    @Autowired

    privateBCryptPasswordEncoderpasswordEncoder;

    publicstaticvoidmain(String[] args) {

        SpringApplication.run(Phase4KitchenStoryApplication.class, args);

    }

    @Override

    publicvoidrun(String... args) throwsException {

        AdminCredcred=newAdminCred();

        cred.setUsername("admin@kitchenstory.com");

        cred.setPassword(this.passwordEncoder.encode("admin123"));

        cred.setRole("ROLE\_ADMIN");

        this.credRepo.save(cred);

    }

}

Pom.xml

<?xml version="1.0" encoding="UTF-8"?>

<projectxmlns="http://maven.apache.org/POM/4.0.0"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <parent>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-starter-parent</artifactId>

        <version>2.7.6</version>

        <relativePath/><!-- lookup parent from repository -->

    </parent>

    <groupId>com.kitchenstory</groupId>

    <artifactId>Phase4\_Kitchen\_Story</artifactId>

    <version>0.0.1-SNAPSHOT</version>

    <name>Phase4\_Kitchen\_Story</name>

    <description>Phase 4 Kitchen Story Backend Project</description>

    <properties>

        <java.version>17</java.version>

    </properties>

    <dependencies>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-data-mongodb</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-security</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-thymeleaf</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-web</artifactId>

        </dependency>

        <dependency>

            <groupId>org.thymeleaf.extras</groupId>

            <artifactId>thymeleaf-extras-springsecurity5</artifactId>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-devtools</artifactId>

            <scope>runtime</scope>

            <optional>true</optional>

        </dependency>

        <dependency>

            <groupId>org.springframework.boot</groupId>

            <artifactId>spring-boot-starter-test</artifactId>

            <scope>test</scope>

        </dependency>

        <dependency>

            <groupId>org.springframework.security</groupId>

            <artifactId>spring-security-test</artifactId>

            <scope>test</scope>

        </dependency>

        <!-- https://mvnrepository.com/artifact/io.jsonwebtoken/jjwt -->

        <dependency>

            <groupId>io.jsonwebtoken</groupId>

            <artifactId>jjwt</artifactId>

            <version>0.9.0</version>

        </dependency>

        <!-- https://mvnrepository.com/artifact/javax.xml.bind/jaxb-api -->

        <dependency>

            <groupId>javax.xml.bind</groupId>

            <artifactId>jaxb-api</artifactId>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <plugin>

                <groupId>org.springframework.boot</groupId>

                <artifactId>spring-boot-maven-plugin</artifactId>

            </plugin>

        </plugins>

    </build>

</project>