**package com.Day1;**

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

**public interface UserRepository extends JpaRepository<User, Integer>{**

**User findByUsernameAndPassword(String username, String password);**

**User findByUsername(String username);**

**}**

package com.Day1;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class FoodStoryApplication {

public static void main(String[] args) {

SpringApplication.run(FoodStoryApplication.class, args);

}

}

**server.port=1020**

**spring.mvc.view.prefix=/WEB-INF/jsps/**

**spring.mvc.view.suffix=.jsp**

**spring.datasource.url=jdbc:mysql://localhost:3306/testdb4**

**spring.datasource.username=root**

**spring.datasource.password=123456789**

**spring.jpa.show-sql=true**

**spring.jpa.hibernate.ddl-auto=update**

**spring.devtools.livereload.enabled=true**

**package com.Day1;**

**public class LoginDto {**

**private String username;**

**private String password;**

**public LoginDto() {**

**// TODO Auto-generated constructor stub**

**}**

**public LoginDto(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;**

**}**

**}**

package com.Day1;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String username;

private String password;

private String email;

private String role;

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

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 getRole() {

return role;

}

public void setRole(String role) {

this.role = role;

}

}

package com.Day1;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/users")

public class UserController {

@Autowired

private UserService userService;

@PostMapping

public ResponseEntity<User> createUser(@RequestBody User user) {

User createdUser = userService.createUser(user);

return new ResponseEntity<>(createdUser, HttpStatus.CREATED);

}

@GetMapping("/{id}")

public ResponseEntity<User> getUserById(@PathVariable int id) {

User user = userService.getUserById(id);

if (user != null) {

return new ResponseEntity<>(user, HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@GetMapping("/username/{username}")

public ResponseEntity<User> getUserByUsername(@PathVariable String username) {

User user = userService.getUserByUsername(username);

if (user != null) {

return new ResponseEntity<>(user, HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@PutMapping("/{id}")

public ResponseEntity<User> updateUser(@PathVariable int id, @RequestBody User user) {

User existingUser = userService.getUserById(id);

if (existingUser != null) {

user.setId(id);

User updatedUser = userService.updateUser(user);

return new ResponseEntity<>(updatedUser, HttpStatus.OK);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteUser(@PathVariable int id) {

User existingUser = userService.getUserById(id);

if (existingUser != null) {

userService.deleteUser(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

}

**package com.Day1;**

**public class UserDto {**

**private String username;**

**private String password;**

**private String email;**

**private String role;**

**public UserDto() {**

**// TODO Auto-generated constructor stub**

**}**

**public UserDto(String username, String password, String email, String role) {**

**super();**

**this.username = username;**

**this.password = password;**

**this.email = email;**

**this.role = role;**

**}**

**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 getEmail() {**

**return email;**

**}**

**public void setEmail(String email) {**

**this.email = email;**

**}**

**public String getRole() {**

**return role;**

**}**

**public void setRole(String role) {**

**this.role = role;**

**}**

**}**

**package com.Day1;**

**public interface UserService {**

**User createUser(User user);**

**User getUserById(int id);**

**User getUserByUsername(String username);**

**User updateUser(User user);**

**void deleteUser(int id);**

**}**

package com.Day1;

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

import org.springframework.stereotype.Service;

@Service

public class UserServiceImpl implements UserService {

@Autowired

private UserRepository userRepository;

@Override

public User createUser(User user) {

return userRepository.save(user);

}

@Override

public User getUserById(int id) {

return userRepository.findById(id).orElse(null);

}

@Override

public User getUserByUsername(String username) {

return userRepository.findByUsername(username);

}

@Override

public User updateUser(User user) {

return userRepository.save(user);

}

@Override

public void deleteUser(int id) {

userRepository.deleteById(id);

}

}

<div class="container">

    <h2>Your Cart</h2>

    <table class="table">

      <thead>

        <tr>

          <th>Item</th>

          <th>Quantity</th>

          <th>Price</th>

          <th>Total</th>

          <th>Actions</th>

        </tr>

      </thead>

      <tbody>

        <!-- Loop through cart items and display details -->

        <tr \*ngFor="let cartItem of cartItems">

          <td>{{ cartItem.food.name }}</td>

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

          <td>{{ cartItem.food.price | currency }}</td>

          <td>{{ cartItem.price | currency }}</td>

          <td>

            <button (click)="changeQuantity(cartItem.food.id, cartItem.quantity - 1)">-</button>

            <button (click)="changeQuantity(cartItem.food.id, cartItem.quantity + 1)">+</button>

            <button (click)="removeFromCart(cartItem.food.id)">Remove</button>

          </td>

        </tr>

      </tbody>

    </table>

    <div class="total">

      <p>Total Price: {{ totalCartPrice | currency }}</p>

      <button (click)="clearCart()">Clear Cart</button>

      <!-- Add checkout button here with appropriate functionality -->

    </div>

  </div>

<div class="container">

    <img [src]="food.imageUrl" [alt]="food.name">

    <div class="details">

        <div class="header">

            <span class="name">

                {{food.name}}

            </span>

            <span class="favorite {{food.favorite?'':'not'}}">❤️</span>

        </div>

        <div class="origins">

            <span \*ngFor="let origin of food.origins">

                {{origin}}

            </span>

        </div>

        <div class="tags">

            <a \*ngFor="let tag of food.tags" routerLink="/tag/{{tag}}">

                {{tag}}

            </a>

        </div>

        <div class="cook-time">

            <span>

                Time to Cook about <strong>{{food.cookTime}}</strong> minutes

            </span>

        </div>

        <div class="price">

            <span>

                {{food.price | currency}}

            </span>

        </div>

<button (click) = "addToCart()">Add to cart</button>

    </div>

</div>

<app-search></app-search>

<app-tags></app-tags>

<ul>

    <li \*ngFor="let food of foods">

      <a routerLink="/food/{{food.id}}">

        <img [src]="food.imageUrl" [alt]="food.name"/>

        <div class="content">

          <div class="name">

            {{food.name}}

          </div>

          <span class="favorite {{food.favorite?'':'not'}}">

            ❤️

          </span>

          <div class="stars">

            <!-- <star-rating

            [value]="food.stars"

            [totalstars]="5"

            checkedcolor="red"

            uncheckedcolor="black"

            size="22px"

            [readonly]="true">

          </star-rating> -->

          </div>

          <div class="product-item-footer">

            <div class="origins">

              <span \*ngFor="let origin of food.origins">

                {{origin}}

              </span>

            </div>

            <div class="cook-time">

              <span>🕒</span>

              {{food.cookTime}}

            </div>

          </div>

          <div class="price">

            <span>

              {{food.price | currency}}

            </span>

          </div>

        </div>

      </a>

    </li>

  </ul>

<header>

    <div class="container">

        <a routerLink="/" class="logo">Kitchen Story 🏠</a>

        <nav>

            <ul>

                <li>

                    <a routerLink="/login">login</a>

                </li>

                <li class="menu-container">

                    <a routerLink="/dashboard">Omi</a>

                    <div class="menu">

                        <a routerLink="/profile">Profile</a>

                        <a routerLink="/Orders">Orders</a>

                        <a>Logout</a>

                    </div>

                </li>

                <li>

                    <a routerLink="/cart-page">

                        Cart <span>{{ cartItemCount }}</span>

                    </a>

                </li>

            </ul>

        </nav>

    </div>

</header>

<div>

    <input #s type="text" placeholder="Search food here!!"

    (keyup.enter)="search(s.value)"

    [value]="searchTerm" />

    <button (click)="search(s.value)">Search</button>

</div>

<div \*ngIf="tags">

<a \*ngFor="let tag of tags" routerLink = "/tag/{{tag.name}}">

    {{tag.name}}({{tag.count}})

</a>

</div>

export class Food{

    id!:string;

    name!:string;

    price!:number;

    tags?:string[];

    favorite!:boolean;

    stars!:number;

    imageUrl!:string;

    origins!:string[];

    cookTime!:string;

}

import { CartItem } from "./CartItem";

export class Cart{

items:CartItem[]=[];

totalPrice:number=0;

totalCount:number=0;

}

import { Food } from "./models/Food";

export class CartItem{

    constructor(public food:Food)

    {

        this.food=food;

    }

    quantity:number=1;

    price:number=this.food.price;

}

export class Tag{

    name!:string;

    count!:number;

}

<app-header></app-header>

<router-outlet></router-outlet>

import {Food} from './app/shared/models/Food';

import { Tag } from './app/shared/tag';

export const sample\_foods: Food[] = [

    {

        id:'1',

        name:'Pizza Pepperoni',

cookTime:'10-20',

price:10,

favorite:false,

origins:['italy'],

stars: 4.5,

imageUrl: 'assets/pizza-2776188.jpg',

tags: ['FastFood','Pizza', 'Lunch'],

    },

    {

        id:'2',

        name:'Meatball',

cookTime:'20-30',

price:20,

favorite:true,

origins:['italy'],

stars: 4.7,

imageUrl: 'assets/manchurian-7025656.jpg',

tags: ['SlowFood', 'Lunch'],

    },

    {

        id:'3',

        name:'Hamburger',

cookTime:'10-15',

price:5,

favorite:false,

origins:['germany','us'],

stars: 3.5,

imageUrl: 'assets/hamburger-1238246.jpg',

tags: ['FastFood','Hamburger'],

    },

    {

        id:'4',

        name:'Fried Patatoes',

cookTime:'15-20',

price:2,

favorite:true,

origins:['belgium','france'],

stars: 3.3,

imageUrl: 'assets/fast-food-1839052.jpg',

tags: ['FastFood','Fry'],

    },

    {

        id:'5',

        name:'Chicken Soup',

cookTime:'40-50',

price:11,

favorite:false,

origins:['india','asia'],

stars: 3.0,

imageUrl: 'assets/potato-soup-2152254.jpg',

tags: ['SlowFood','Soup'],

    },

    {

        id:'6',

        name:'Vegitable Pizza',

cookTime:'40-50',

price:9,

favorite:false,

origins:['italy'],

stars: 4.0,

imageUrl: 'assets/pizza-5179939.jpg',

tags: ['FastFood','Pizza', 'Lunch'],

    },

]

export const sample\_tags:any=[

  {name: 'All', count:6},

  {name:'FastFood',count:4},

  {name:'Pizza', count:2},

  {name: 'Lunch',count:3},

  {name: 'SlowFood',count:2},

  {name:'Hamburger', count:1},

  {name: 'Fry', count:1},

  {name: 'Soup', count:1}

]

.container{

    display: flex;

    justify-content: center;

    align-items: center;

    flex-wrap: wrap;

    margin: 3rem;

}

.container > \*{

    min-width:25rem;

    max-width: 40rem;

}

img{

    border-radius: 3rem;

    flex: 1 0;

    object-fit: cover;

    height:35 rem;

    margin: 1rem;

}

.details{

    width: 100%;

    display: flex;

    flex-direction: column;

    flex: 1 0;

    border: 3rem;

    padding: 2rem;

    color: black;

    margin-left: 1 rem;

}

.header{

    display: flex;

    justify-content: space-between;

}

.name{

    font-size: 3rem;

    font-weight: bold;

}

.favorite{

    color: red;

    font-size: 2.5rem;

}

.favorite.container{

    color: grey;

}

.origins{

    display: flex;

    flex-wrap: wrap;

    margin: 1rem 0;

}

.origins span{

    padding: 0.5rem;

    font-size: 1.2rem;

    margin: 0.5rem 0.5rem 0.5rem 0;

    border-radius: 2rem;

    background-color: aliceblue;

}

.tags{

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

}

tags a{

    background-color: #f0f0f0;

    padding: 0.3rem 1rem;

    margin: 0.2rem 0.15rem;

    border-radius: 10rem;

    font-weight: 600;

    color: blue;

}

cook-time{

    margin-top: 1rem;

}

.cook-time span{

    padding: 0.6rem 2rem 0.6rem 0;

    border-radius: 10rem;

    font-size: 1.3rem;

}

.price{

    font-size: 1.8rem;

    margin: 2rem 2rem 2rem 0;

    color: green;

}

.price::before{

    content: "price:";

    color: darkgrey;

}

button{

    color: white;

    background-color: red;

    border: none;

    font-size: 1.2rem;

    padding: 1rem;

    border-radius: 10rem;

    outline: none;

}

button:hover{

    opacity: 0.9;

    cursor: pointer;

}

ul{

    display: flex;

    justify-content: center;

    align-items: center;

    flex-wrap: wrap;

    list-style-type: none;

    padding: 0;

  }

  li a{

    height: 22.5rem;

    width: 20rem;

    border: 1px solid whitesmoke;

    border-radius: 1rem;

    margin: 0.5rem;

    display: flex;

    flex-direction: column;

    overflow: hidden;

    color: #e72929;

  }

  li a:hover{

    opacity: 0.9;

    cursor: pointer;

  }

  li a img{

    object-fit: cover;

    height: 14.5rem;

  }

  .content{

    margin-top: 0.3rem;

    padding: 0.5rem 1rem;

    position: relative;

    height: 7rem;

  }

  .favorite{

    position: absolute;

    top: 0.5rem;

    right: 1rem;

    font-size: 1.2rem;

  }

  .favorite.not{

    color: grey;

  }

  .stars{

    margin-bottom: 0.2rem;

  }

  .product-item-footer{

    display: flex;

    justify-content: space-between;

    align-items: flex-start;

  }

  .origins{

    flex: 9;

  }

  .origins span{

    border-radius: 2rem;

    background-color: whitesmoke;

    display: inline-block;

    font-size: 0.8rem;

    color: grey;

    margin-right: 0.3rem;

    padding: 0 0.5rem;

    margin-top: 0.2rem;

  }

  .cook-time{

    font-size: 0.9rem;

    flex: 3;

    text-align: right;

  }

  .price{

    position: absolute;

    bottom: 0;

    left: 0.9rem;

    background: white;

    padding: 0.3rem 100% 0 0;

    color: #414141;

    margin-top: 0.7rem;

    font-size: 1.1rem;

  }

header{

    position: relative;

    background: white;

    padding: 0;

    border-bottom: 1px solid red;

}

a{

    color: red;

}

a:hover{

    background-color: red;

    color: white;

    cursor: pointer;

}

.container{

    margin: 0 auto;

    display: flex;

    justify-content: space-between;

}

a.logo{

    font-weight: bold;

    padding: 1rem;

}

ul{

    display: flex;

    list-style-type: none;

    margin: 0;

}

ul a{

    padding: 1rem;

    display: inline-block;

}

a span{

    background-color: red;

    color: white;

    padding: 0.1rem 0.45rem;

    border-radius: 100rem;

    font-size: 0.9rem;

}

.menu-container{

    position: relative;

}

.menu{

    position: absolute;

    z-index: 1;

    background: whitesmoke;

    display: none;

}

.menu-container:hover .menu{

    display: block;

}

.menu a{

    width: 100%;

    min-width: 8rem;

}

div{

    display: flex;

    flex-wrap: wrap;

    justify-content: center;

}

div a{

    background-color: #f0f0f0;

    padding: 0.3rem;

    margin: 0.2rem 0.15rem;

    border-radius: 10rem;

    font-weight: 600;

    color: blue;

}

/\* Styles for the search container \*/

div {

    display: flex;

    justify-content: center;

    margin-top: 3rem;

    margin-bottom: 1.5rem;

  }

  /\* Styles for the search input \*/

  input {

    padding: 1.2rem;

    height: 3rem;

    width: 20rem;

    background-color: #f1f1f1;

    font-size: 1.1rem;

    font-weight: 500;

    outline: none;

    border: none;

    border-radius: 10rem 0 0 10rem;

  }

  /\* Styles for the search button \*/

  div button {

    color: grey;

    height: 3rem;

    width: 5rem;

    font-size: 1rem;

    border-radius: 0 10rem 10rem 0;

    border: none;

    background-color: red;

    color: white;

    opacity: 0.8;

    outline: none;

  }

  /\* Styles for the search button on hover \*/

  div button:hover {

    opacity: 1;

    cursor: pointer;

  }