<!-- 引入外部CSS样式表文件stylesheet.css，用于定义页面的整体样式 -->

<link rel="stylesheet" href="./css/stylesheet.css">

<!-- 引入Swiper轮播图插件的CSS样式文件，用于实现轮播图相关的样式效果 -->

<link rel="stylesheet" href="./js/swiper.min.css">

<style>

/\* 定义类名为.swiper-container的元素样式，这里用于设置轮播图容器的样式。

将宽度设置为占满父元素的100%，高度固定为460px，确定轮播图在页面中的尺寸大小 \*/

.swiper-container {

width: 100%;

height: 460px;

}

/\* 定义类名为.swiper-slide img的元素样式，即轮播图中每个幻灯片里图片的样式。

display: block; 将图片设置为块级元素，使其能独占一行，方便设置宽度、高度等属性。

width: 100%; 和 height: 100%; 让图片的宽度和高度都与幻灯片尺寸一致，占满整个幻灯片空间。

object-fit: cover; 规定图片按比例缩放，以覆盖整个幻灯片区域，可能会裁剪图片部分内容，确保图片能完整填充幻灯片且保持合适比例显示 \*/

.swiper-slide img {

display: block;

width: 100%;

height: 100%;

object-fit: cover;

}

/\* 定义类名为.middder的元素样式，通过display: flex; 将该元素设置为弹性盒子布局，使其内部子元素可以按照弹性布局规则进行排列。

flex-direction: column; 明确了弹性盒子内元素的排列方向为垂直方向（列方向），子元素会从上到下依次排列 \*/

.middder {

display: flex;

flex-direction: column;

}

/\* 定义类名为.zhushijingxuan的元素样式，同样先通过display: flex; 将该元素设置为弹性盒子布局，方便对子元素进行灵活排列。

flex-wrap: wrap; 规定当弹性盒子中的子元素超出一行的宽度时，自动换行排列，使得子元素能够在多行中展示，避免超出容器宽度范围 \*/

.zhushijingxuan {

display: flex;

flex-wrap: wrap;

}

</style>

</head>

<body>

<!-- 头部区域，用于展示页面的标题等基本信息， <header>

<h1>欢迎来到美食的奇妙世界</h1>

<p>您将在盛宝金融科技商学院美食城开启一场美食之旅</p>

</header

<!-- 显示网站的logo图片，若图片无法正常显示则显示alt属性中的替代文本“美食世界” -->

<img src="./css/图片/7482945896bdcf34e1c2f42b3785cde.png" alt="美食世界">

<span>美食世界</span>

</div>

<ul class="nav-links">

<!-- 首页导航链接，添加了active类名可能用于表示当前选中状态，点击可进行页面跳转 -->

<li><a href="#" class="active">首页</a></li>

<li class="dropdown">

<!-- 菜谱下拉菜单按钮，点击可展开包含不同菜谱分类链接的下拉内容 -->

<a href="#" class="dropbtn">菜谱 <span class="arrow-down">&#9660;</span></a>

<div class="dropdown-content">

<ul>

<!-- 家常菜菜谱分类链接，点击跳转到对应的外部网页（https://hom

下面是我们小组做的跳转链接，主要是有菜谱、食材、川菜、云贵菜这几个页面作为展示，现在给大家做一下演示》

<!-- 搜索框，用户可输入关键词来搜索菜谱、菜单、食材、用户等内容 -->

<input type="text" placeholder="搜索菜谱、菜单、食材、用户">

<button><img src="./css/图片/4eb062a5a8dd1b0de63e8a3e959bc6f.png" alt="搜索"></button>

</div>

<div class="user-actions">

<!-- 发布按钮，类型为submit，可能用于提交用户发布的内容（需结合后端功能实现具体发布逻辑） -->

<button type="submit" class="clickable-btn">发布</button>

<!-- 登录/注册按钮，点击后通过JavaScript改变页面位置跳转到登录页面（此处/login路径需后端配合实现功能） -->

<button onclick="location.href='/login'" class="clickable-btn">登录/注册</button>

</div>

</nav>

<!-- 内容区域 -->

<!-- 左侧导航栏和图片轮播相关内容，由郭瑾涵负责 -->

<div class="main-container">

<div class="box">

<div class="box1">

<div class="pic">《下面的这些部分，也是跟上面的一样，由img标签来显示相应图片》

《读一下英文》

<div class="swiper-container">

<div class="swiper-wrapper">《这里也是运用了JS插件，实现了轮播图的正常演示功能》

<!-- 轮播图的每个幻灯片，展示不同美食图片，此处展示沙拉图片，若图片无法正常显示则显示alt属性中的替代文本“11” -->

<div class="swiper-slide"><img src="./css/图片/沙拉e.jpg" alt="11"></div>

<!-- 展示黄焖鸡图片，若图片无法正常显示则显示alt属性中的替代文本“22” -->

<div class="swiper-slide"><img src="./css/图片/黄焖鸡z2.jpg" alt="22"></div>

<!-- 展示鸡爪煲图片，若图片无法正常显示则显示alt属性中的替代文本“33” -->

<div class="swiper-slide"><img src="./css/图片/鸡爪煲.jpg" alt="33"></div>

<!-- 展示沙拉图片，若图片无法正常显示则显示alt属性中的替代文本“44” -->

<div class="swiper-slide"><img src="./css/图片/沙拉q.jpg" alt="44"></div>

<!-- 展示沙拉图片，若图片无法正常显示则显示alt属性中的替代文本“55” -->

<div class="swiper-slide"><img src="./css/图片/沙拉w.jpg" alt="55"></div>

</div>

<div class="swiper-pagination"></div>

</div>

</div>

<script src="./js/swiper.min.js"></script>

<script>

// 页面加载完成后执行的函数，用于初始化Swiper轮播图插件，设置轮播图的各种属性和效果

window.onload = function () {

var swiper = new Swiper('.swiper-container', {

// 设置轮播图循环播放，即播放到最后一张后会自动回到第一张继续播放

loop: true,

// 设置自动播放间隔为2000毫秒（2秒），使轮播图自动切换幻灯片

autoplay: 2000,

pagination: {

el: '.swiper-pagination',

// 设置轮播图分页器可点击切换幻灯片，用户点击分页器上的小点可以切换到对应的幻灯片

clickable: true,

},

grabCursor: true,

effect: "coverflow",

centeredSlides: true,

coverflowEffect: {

rotate: 20,

stretch: 0,

depth: 80,

modifier: 1,

slideShadows: true,

},

});

};

</script>

<!-- 空的区域，防止被图片覆盖，由王润姿添加 -->

<div style="height: 20px;"></div>

<!-- 特色美食和菜谱区域，由王润姿、李睿负责 -->

<div class="recipe-list">

<div class="recipe-card">

<p>今日推荐指数--五颗星</p>

<!-- 显示对应菜谱的图片，若图片无法正常显示则显示alt属性中的替代文本“菜谱 1 图片” -->

<img src="./css/图片/19361376\_144045227000\_2.jpg" alt="菜谱 1 图片">

<h3>花菇鸡饭</h3>

<!-- 点击按钮调用addToCart函数，传递菜品相关信息（菜品id、名称、价格、图片路径），用于将菜品添加到购物车 -->

<button onclick="addToCart(1, '花菇鸡饭', 25, './css/图片/19361376\_144045227000\_2.jpg')">加入购物车</button>

</div>

<div class="recipe-card">

<p>今日推荐指数--四颗星</p>

<img src="./css/图片/v2-0390e813381771812a9ba5a706c3d137\_r.jpg" alt="菜谱 2 图片">

<div class="cart"> <!-- 这个 div 用于包裹整个购物车相关的内容，起到容器的作用 -->

<p class="gouwuche">购物车</p>

<!-- 显示“购物车”字样，用于标识这部分内容是购物车区域，class="gouwuche" 可能用于后续的样式设置 -->

<div id="cart-items">

<!-- 这个 div 是用来动态添加购物车商品项的容器，id="cart-items" 方便通过 JavaScript 操作来添加、移除或修改里面的内容 -->

<!-- 购物车商品项将动态添加在此处 -->

</div>

<div class="cart-total">

<!-- 这个 div 用于展示购物车总价相关的内容 -->

Total: $<span id="total-amount">0</span>

<!-- 显示“Total: $”字样表示总价，后面的 <span> 标签用于通过 JavaScript 动态更新总价金额，id="total-amount" 方便操作 -->

</div>

<button style="margin-left: 80px;" onclick="checkout()">结算</button>

<!-- 结算按钮，点击后会调用 checkout() 函数来执行购物车结算相关的逻辑，通过 style 属性设置了左边距为 80px，用于调整按钮在页面中的位置 -->

</div>

</div>

<!-- 购物车区域——王润姿 -->

<script>

// 以下是 JavaScript 代码部分，用于实现购物车相关的功能逻辑

// 模拟商品数据，定义一个数组，每个元素是一个对象，代表一种商品，包含商品的 id、名称、价格、图片路径等信息

const products = [

{ id: 1, name: '花菇鸡饭', price: 25, image: './css/图片/19361376\_144045227000\_2.jpg' },

{ id: 2, name: '兰州拉面', price: 18, image: './css/图片/v2-0390e813381771812a9ba5a706c3d137\_r.jpg' },

{ id: 3, name: '北京烤鸭', price: 35, image: './css/图片/OIP-C (1).jpg' },

{ id: 4, name: '水煮鱼', price: 45, image: '../css/图片/沙拉e.jpg' },

{ id: 4, name: '黄焖鸡', price: 25, image: '../css/图片/黄焖鸡z2.jpg' },

];

// 用于存储购物车中的商品，初始化为空数组，后续会根据用户操作添加商品对象进来

let cart = [];

// 渲染购物车商品项的函数，用于将购物车中的商品信息显示在页面上对应的区域（即 id 为 "cart-items" 的 div 中）

function renderCartItems() {

const cartItemsDiv = document.getElementById('cart-items');

// 先清空 "cart-items" div 内原有的内容，避免重复添加造成混乱

cartItemsDiv.innerHTML = '';

cart.forEach(item => {

const cartItemDiv = document.createElement('div');

// 为创建的每个商品项 div 添加 "cart-item" 类名，方便后续通过 CSS 进行样式设置

cartItemDiv.classList.add('cart-item');

cartItemDiv.innerHTML = `

<img src="${item.image}" alt="${item.name}">

<span>${item.name} - $${item.price}</span>

<button onclick="removeFromCart(${item.id})">移除</button>

`;

// 将构建好的包含商品信息和移除按钮的 div 添加到 "cart-items" div 中，实现商品项的动态添加显示

cartItemsDiv.appendChild(cartItemDiv);

});

// 调用更新总价的函数，确保总价在商品项变化后能及时更新显示正确金额

updateTotal();

}

// 添加商品到购物车的函数，接收商品的 id、名称、价格、图片路径等参数

function addToCart(productId, productName, price, image) {

// 判断购物车中商品数量是否已经达到 3 个，如果是则弹出提示，不允许再添加商品

if (cart.length >= 3) {

alert('购物车最多只能添加三种不同的食物！');

return;

}

const product = products.find(p => p.id === productId);

// 在购物车已有的商品中查找是否已经存在当前要添加的商品（通过 id 判断）

const existingItem = cart.find(item => item.id === productId);

if (existingItem) {

// 如果商品已存在，将其数量加 1，表示增加该商品的购买数量

existingItem.quantity++;

} else {

// 如果商品不存在，创建一个新的商品对象，包含商品的各项信息，并将数量初始化为 1，然后添加到购物车数组中

cart.push({ id: productId, name: productName, price: price, image: image, quantity: 1 });

}

// 添加商品后调用渲染函数，更新购物车商品项在页面上的显示

renderCartItems();

}

// 从购物车移除商品的函数，接收要移除商品的 id 参数

function removeFromCart(productId) {

// 在购物车数组中查找要移除商品的索引位置

const index = cart.findIndex(item => item.id === productId);

if (index > -1) {

// 如果找到了该商品（索引大于 -1），则从购物车数组中移除该商品对象

cart.splice(index, 1);

// 移除后调用渲染函数，更新购物车商品项在页面上的显示，使其反映最新的购物车内容

renderCartItems();

}

}

// 更新购物车总价的函数，用于根据购物车中商品的数量和价格计算并更新显示总价金额

function updateTotal() {

const totalAmountSpan = document.getElementById('total-amount');

let total = 0;

cart.forEach(item => {

// 遍历购物车中的每个商品，将商品价格乘以数量后累加到总价变量中

total += item.price \* item.quantity;

});

// 将计算得到的总价金额保留两位小数后更新到页面上对应显示总价的 <span> 标签中（id 为 "total-amount"）

totalAmountSpan.textContent = total.toFixed(2);

}

// 结算购物车的函数，用于处理购物车结算相关的逻辑，如弹出提示、清空购物车等

function checkout() {

// 判断购物车是否为空，如果为空则弹出提示，不执行结算操作

if (cart.length === 0) {

alert('购物车为空！');

return;

}

// 这里可以添加实际的结算逻辑，如与后端交互，发送购物车数据、生成订单等（目前只是简单弹出提示模拟结算成功）

alert('结算成功，共' + cart.length + '件商品，总价：$' + document.getElementById('total-amount').textContent);

// 结算完成后清空购物车数组

cart = [];

// 调用渲染函数，更新购物车商品项在页面上的显示，此时购物车为空，页面显示也会相应更新为空

renderCartItems();

}

</script>

<!-- 丰富菜品 郭瑾涵-->

<div class="middder">

<!-- 这个 div 使用了类名 "middder"，可能通过 CSS 样式设置了弹性盒子相关的布局属性，整体用于包裹下面的菜品分类内容 -->

<div>

<!-- 这里的内部 div 用于区分不同的菜品分类板块，此为第一个板块 -->

<h1>肉食精选</h1>

<!-- 显示“肉食精选”标题，用于标识下面展示的菜品是肉食类的精选菜品 -->

<div class="classList">

<!-- 这个 div 可能用于对菜品列表进行样式设置，作为菜品列表的容器 -->

《下面的部分跟这里的都是一样的，就跟大家略过了》

<!-- 饮食百科——王润姿，郭瑾涵 -->

<div class="container"> 《这里我们也是定义了一个盒子来作为下面内容的一个容器》

<h1>饮食百科</h1>

<!-- 显示“饮食百科”标题，用于标识这部分内容是关于饮食相关知识的百科板块 -->

<div class="category-box">

<!-- 这个 div 可能作为整个饮食百科分类内容的容器，用于包裹下面不同分类的内容 -->

<div class="category">

<!-- 每个 <div class="category"> 代表饮食百科的一个分类板块，这里是第一个分类板块 -->

<h2>烹饪技巧</h2>

<!-- 显示“烹饪技巧”标题，用于标识这个分类板块的主题是烹饪方面的技巧知识 -->

<img src="./css/图片/饮食百科.jpg" alt="Cooking Image">

<!-- 显示一张与烹饪技巧相关的图片，若图片无法正常显示则显示 alt 属性中的替代文本“Cooking Image” -->

<ul>

<!-- 无序列表，用于罗列该分类下的具体知识点 -->

<li>无骨鸡爪的制作方法</li>

<li>详解4个对鸡肉营养认识的误区</li>

<li>鸡蛋怎么煮好吃</li>

<li>如何快速清洗葡萄</li>

<li>让老北京水爆肚好吃的关键技巧</li>

</ul>

</div>

<div style="height: 525px;"></div>

<!-- 这个空的 div 通过设置高度为 525px，可能用于在页面布局中起到间隔、留白等作用，具体根据整体页面设计需求而定 -->

<!-- 底部区域——李宗旭 -->

<footer>

<!-- HTML 页面的底部区域，通常用于放置版权信息、联系方式、网站相关链接等内容 -->

<div class="footer-links">

<!-- 这个 div 用于罗列一些链接相关的文本，如“关于我们”“联系我们”等，可能后续会通过 CSS 样式设置为可点击的链接或添加相关交互逻辑 -->

关于我们

联系我们

隐私政策

使用条款

</div>

<div class="contact-info">

<!-- 这个 div 用于展示联系相关的信息，如联系电话和电子邮箱 -->

联系电话: 123 - 456 - 7890

电子邮箱: info@example.com

</div>

<p>&copy;美食的奇妙世界。保留所有权利。</p>

<!-- 显示版权声明信息，&copy; 是版权符号，表明该页面的版权归属 -->

</footer>

</body>

</html>