

BLM2537

Site Yapma Ödevi

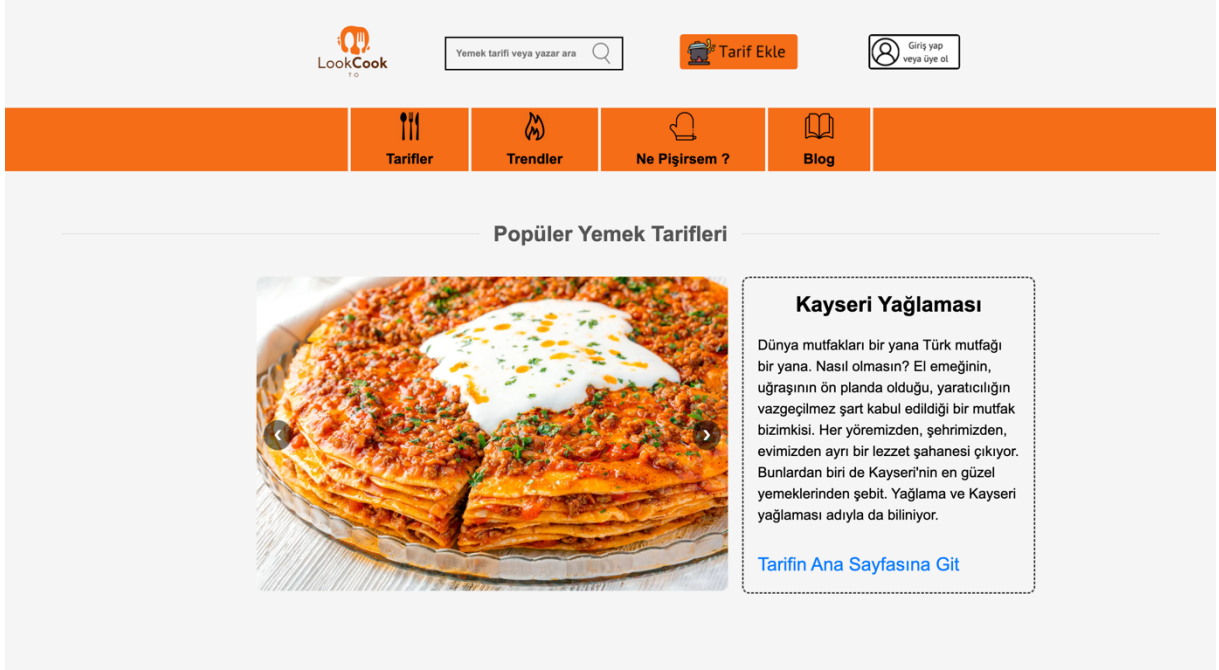
Semih TAKILAN 23291251

Github hesabı : semihtakilan

Proje git linki: <https://github.com/semihtakilan/LooktoCook>

Proje Sunum Linki: https://drive.google.com/file/d/10QUQwVMhXyz-zF0C8d7OO_bQ2FrNAmlk/view?usp=sharing

Sitemin Ana Sayfası



Bu kısmı oluşturmak için Header, Navbar, Content, En Yeniler ve Footer olmak üzere 5 adet ana div kullandım.

```

<!--Header-->
<div class="header">
  <a href="index.html">
    
  </a>

  <form action="tabs/ara.html" method="GET" class="search_bar">
    <input class="search-bar" type="text" name="q" placeholder="Yemek tarifi veya yazar ara" required>
    
  </form>
  <a href="tabs/post.html">
    
  </a>

  <div id="user" class="user">
    <a href="tabs/login/login.php">
      
    </a>
  </div>
</div>

<!--Navigation Bar-->
<div class="topnav">
  <a href="tabs/recipes.html">
    
    <span><b>Tarifler</b></span>
  </a>
  <a href="tabs/trends.html">
    
    <span><b>Trendler</b></span>
  </a>
  <a href="tabs/random.html">
    
    <span><b>Ne Pişirsem ?</b></span>
  </a>
  <a href="tabs/blog.html">
    
    <span><b>Blog</b></span>
  </a>
</div>

<!--Content-->
<div class="content-header">
  <span>Popüler Yemek Tarifleri</span>
</div>
<div class="container">
  <!-- Sol Taraf: Yemek Fotoğrafı ve Oklar -->
  <div class="image-container">
    <button class="nav-btn prev-btn" onclick="changeSlide(-1, recipes)"></button>
    
    <button class="nav-btn next-btn" onclick="changeSlide(1, recipes)"></button>
  </div>

  <!-- Sağ Taraf: Tarif Metni -->
  <div class="recipe-details" id="recipeDetails">
    <h2 id="recipeTitle">Yemek Tarifi</h2>
    <p id="recipeDescription">Burada yemeğin tarifi yer alacak...</p>
    <a href="index.html" id="recipeLink" class="recipe-link">Tarifin Ana Sayfasına Git</a>
  </div>
</div>

<!--En Yeniler-->
<div class="content-header">
  <span>En Yeniler</span>
</div>
<div class="cards-container" id="cards-container"></div>

<!--Footer-->
<footer>
  <div class="footer-container">
    <div class="footer-nav">
      <h3>Site Haritası</h3>
      <ul>
        <li><a href="index.html">Ana Sayfa</a></li>
        <li><a href="./tabs/info.html">Hakkımızda</a></li>
        <li><a href="./tabs/contact.html">Bize Ulaşın</a></li>
        <li><a href="#">Gizlilik Politikası</a></li>
      </ul>
    </div>
  </div>
</footer>

```

Ana sayfamı açıklarken bölüm bölüm gidiyim, öncelikle header ve navbar kısmını basit bir şekilde html vs css ile düzenledim.

Content kısmında da Sol tarafta tarifi görseli sağda ise açıklaması ve tarif sayfasına yönlendiren link olacak şekilde bir div oluşturup css kodlarıyla istediğim şekli verdim.

```
let currentIndex = 0;
let recipes = [];

const foodImage = document.getElementById("foodImage");
const recipeTitle = document.getElementById("recipeTitle");
const recipeDescription = document.getElementById("recipeDescription");

fetch('./datas/populer.json')
  .then(response => {
    if (!response.ok) {
      throw new Error('JSON dosyası yüklenemedi!');
    }
    return response.json();
  })
  .then(data => {
    recipes = data;
    changeSlide(0);
  })
  .catch(error => {
    console.error('Hata:', error);
  });

function changeSlide(direction) {
  currentIndex += direction;

  if (currentIndex < 0) {
    currentIndex = Object.keys(recipes).length - 1;
  } else if (currentIndex >= Object.keys(recipes).length) {
    currentIndex = 0;
  }

  foodImage.src = recipes[currentIndex].image;
  recipeTitle.textContent = recipes[currentIndex].title;
  recipeDescription.innerHTML = recipes[currentIndex].description;
  const recipeLink = document.getElementById("recipeLink");
  recipeLink.href = recipes[currentIndex].link;
}
```

Burdaki kod parçası content kısmının oluşmasını sağlayan kısım
Öncelikle JSON dosyasındaki yemek tariflerini almak için “fetch”
fonksiyonu kullanıldı. “populer.json” dosyasındaki veriler, başarılı
bir şekilde yüklendikten sonra “recipes” dizisine atanıyor.
Eğer JSON dosyası yüklenemezse, hata mesajı kullanıcıya
gösterilmektedir.

Sonrasında ise okların işlevini yerine getirmesi için “changeSlide”
fonksiyonu, yemek tariflerini değiştiren ana fonksiyondur. Bu
fonksiyon, kullanıcıdan gelen yön (ileri veya geri) bilgisine göre
“currentIndex” değerini günceller.

Eğer “currentIndex“ dizinin sınırlarına ulaşırsa, gösterim döngüsel
olarak başa sarar ya da sona gider.

En son olarak “foodImage”, “recipeTitle”, “recipeDescription” gibi
DOM öğeleri, her yeni yemek tarifine geçiş yapıldığında ilgili
verilerle (görsel, başlık, açıklama) güncellenir.

Ayrıca, yemek tarifinin bağlantısı (“recipeLink”), kullanıcıyı tarifi
detay sayfasına yönlendirmek için dinamik olarak değiştirilir.

```

fetch('./datas/en_yeniler.json')
  .then(response => {
    if (!response.ok) {
      throw new Error('JSON dosyası yüklenemedi!');
    }
    return response.json();
  })
  .then(data => {
    const mainDiv = document.getElementById('cards-container');

    data.forEach(item => {
      const card = document.createElement('div');
      card.className = 'card';

      card.innerHTML = `
        
        <div class="card-info">
          <div class="card-badge">${item.badge}</div>
          <a href="/tabs/recipes/tarifler.html?category=${item.category}">div class="card-category">${item.category}</div></a>
          <h3 class="card-title">${item.title}</h3>
          <p class="card-author">${item.author}</p>
          <div class="card-details">
            <span class="card-time">${item.time}</span>
            <span class="card-rating">${item.rating}</span>
          </div>
        </div>
      `;

      mainDiv.appendChild(card);
    });
  })
  .catch(error => {
    console.error('Hata:', error);
  });
}

```

Bu kod, bir JSON dosyasındaki yemek verilerini alarak, her yemek için bir kart oluşturur ve bu kartları HTML sayfasına ekler. Bu işlemi gerçekleştiren adımlar şöyle:

1. JSON Dosyasını Yükleme:

`fetch('./datas/en_yeniler.json')`: Bu satırda `en_yeniler.json` adlı dosya sunucudan alınmaya çalışılıyor. Bu dosya, yemek verilerini içeriyor.

`.then(response => {...})`: Dosya başarıyla yüklendiyse, veriyi JSON formatına çeviriyoruz. Ama eğer dosya düzgün yüklenmezse, hata mesajı gösteriyoruz.

2. Veri Üzerinde İşlem Yapmak:

JSON dosyasındaki veriler başarıyla alındıktan sonra, her bir yemek için bir işlem yapıyoruz.

`data.forEach(item => {...})`: Burada, JSON verisindeki her bir yemek (item) için işlem yapıyoruz.

`const card = document.createElement('div');` Her yemek için bir div (kart) oluşturuyoruz.

3. Yemek Kartlarını Oluşturmak:

Her kartın içine:

Yemek görseli (`imgSrc`),

Yemek kategorisi (`category`),

Yemek başlığı (`title`),

Yazar adı (`author`),

Hazırlama süresi ve puanı gibi bilgiler yerleştiriliyor.

Bu bilgiler `card.innerHTML = ...` şeklinde her karta ekleniyor.

4. Kartları Ekrana Eklemek:

`mainDiv.appendChild(card);` Son olarak, her kartı `cards-container` adlı div'in içine ekliyoruz. Yani her yemek kartı, sayfada görünür oluyor.

5. Hata Durumu:

Eğer JSON dosyasında bir problem çıkarsa (mesela dosya yüklenemezse), bu hata `catch` ile yakalanıyor ve konsola yazdırılıyor.

Kodun Amacı:

Bu kod, JSON dosyasındaki yemekleri alıp, her biri için dinamik olarak bir kart oluşturuyor ve bu kartları ekrana ekliyor. Böylece, yemeklerin bilgileri düzenli bir şekilde kullanıcıya sunulmuş oluyor.


Yani, bu sayede:

Yemekler Dinamik Olarak Görünür Oluyor: JSON dosyasına yeni yemekler eklendikçe, bu yemekler otomatik olarak sayfada gösteriliyor.

Kullanıcı Dostu Arayüz: Kartlarda yemek görselleri, başlıkları, kategorileri gibi bilgiler bulunuyor, böylece kullanıcılar daha kolay bir şekilde yemekleri inceleyebiliyor.


Bu tarzı kodlamak için baya uğraştım ama buna değdi ve sitemin çoğu sayfasında kullandım. Mesela trends, tarifler, bugün ne pişirsem sayfalarında bu kodu kullandım tabi filtre işlemine tabi tutarak.

Sitemin Giriş Sistemi



Giriş Yap

Henüz bir hesabın yok mu? [Kayıt ol](#)



Kayıt Ol

Zaten bir hesabın var mı? [Giriş Yap](#)

Burda görüldüğü üzere giriş yap ve kayıt ol kısımlarından oluşuyo bi iki sayfa arasında geçiş yapmak için en altta “Kayıt Ol” ve “Giriş Yap” linkleri ekledim burdaki sistem php ile çalışıyor.

1. Kayıt Olma (register.php)

Kayıt olma sayfasında kullanıcılar, adlarını, e-posta adreslerini ve şifrelerini girerek sisteme üye olabilirler. Sayfada formu doldurduktan sonra veriler “register.class.php” dosyasına gönderilir ve burada işlenir.

İşleyiş:

Form Verisi: Kullanıcı formu doldurup "Kayıt Ol" butonuna bastığında, veriler “\$_POST” ile alınır.

“RegisterUser” Sınıfı: “register.class.php” dosyasında RegisterUser sınıfı, kullanıcının verilerini alır ve doğrulama işlemi yapar.

Veri Temizliği: Kullanıcıdan gelen veriler “sanitizeInput” fonksiyonu ile temizlenir (HTML özel karakterlerinden arındırılır).

Şifre Şifreleme: Kullanıcının şifresi, güvenlik amacıyla “password_hash” fonksiyonu ile şifrelenir.

Kullanıcı Kontrolü: Kullanıcı adı daha önce alınmışsa, hata mesajı gösterilir. Eğer kullanıcı adı benzersizse, veritabanı olarak kullanılan “data.json” dosyasına yeni kullanıcı eklenir.

Başarı ve Hata Mesajları: Kullanıcıya işlem başarılı ya da başarısız olduğuna dair mesajlar gösterilir. register.php ve register.class.php Görselde

```
<?php require("register.class.php") ?>
<?php
    if(isset($_POST['submit'])){
        $user = new RegisterUser(username: $_POST['username'], email: $_POST['email'], password: $_POST['password']);
    }
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Register form</title>
    <link rel="icon" href="../../../images/logo2.png">
</head>
<body>
    <div class="form-container">
        <!-- Logo -->
        <a href="../../../index.html"></a>

        <form action="" method="post" enctype="multipart/form-data" autocomplete="off">
            <h2>Kayıt Ol</h2>
            <input type="text" name="username" placeholder="Kullanıcı Adı">

            <input type="email" name="email" placeholder="E-Posta">

            <input type="password" name="password" placeholder="Şifre">

            <button type="submit" name="submit">Kayıt Ol</button>

            <div>Zaten bir hesabın var mı? <a href="login.php">Giriş Yap</a></div>
            <p class="error"><?php echo @$user->error ?></p>
            <p class="success"><?php echo @$user->success ?></p>
        </form>
    </div>
</body>
</html>
```

```

<?php
1 reference | 0 implementations
class RegisterUser{
4 references
    private $username;
2 references
    private $email;
3 references
    private $raw_password;
2 references
    private $encrypted_password;
5 references
    public $error;
3 references
    public $success;
2 references
    private $storage = "../datas/data.json";
4 references
    private $stored_users;
2 references
    private $new_user;

1 reference | 0 overrides
    public function __construct($username, $email, $password){

        $this->username = $this->sanitizeInput(input: $username);
        $this->email = $this->sanitizeInput(input: $email);
        $this->raw_password = $this->sanitizeInput(input: $password);

        $this->encrypted_password = password_hash(password: $this->raw_password, algo: PASSWORD_DEFAULT);

        $this->stored_users = json_decode(json: file_get_contents(filename: $this->storage), associative: true);

        $this->new_user = [
            "username" => $this->username,
            "email" => $this->email,
            "password" => $this->encrypted_password,
        ];

        if($this->checkFieldValues()){
            $this->insertUser();
        }
    }

3 references
    private function sanitizeInput($input): string {
        $input = $input ?? '';
        return htmlspecialchars(string: trim(string: $input), flags: ENT_QUOTES, encoding: 'UTF-8');
    }

1 reference
    private function checkFieldValues(): bool{
        if(empty($this->username) || empty($this->raw_password)){
            $this->error = "Both fields are required.";
            return false;
        }else{
            return true;
        }
    }

1 reference
    private function usernameExists(): bool{
        foreach($this->stored_users as $user){
            if($this->username == $user['username']){
                $this->error = "Username already taken, please choose a different one.";
                return true;
            }
        }
        return false;
    }

1 reference
    private function insertUser(): string{
        if($this->usernameExists() == FALSE){
            array_push(array: &$this->stored_users, values: $this->new_user);
            if(file_put_contents(filename: $this->storage, data: json_encode(value: $this->stored_users, flags: JSON_PRETTY_PRINT))){
                return $this->success = "Kayıt olma başarılı";
            }else{
                return $this->error = "Something went wrong, please try again";
            }
        }
    }
}

```

2. Giriş Yapma (login.php)

Giriş sayfası, kullanıcıların sistemdeki hesaplarına erişmelerini sağlar. Kullanıcı adı ve şifre girildikten sonra, bilgiler “login.class.php” dosyasına gönderilir ve doğrulama yapılır.

İşleyiş:

Form Verisi: Kullanıcı, giriş yapmak için kullanıcı adı ve şifreyi form aracılığıyla gönderir.

LoginUser Sınıfı: “login.class.php” dosyasındaki “LoginUser” sınıfı, kullanıcının giriş bilgilerini alır ve veritabanındaki bilgilerle karşılaştırır.

Şifre Doğrulama: Kullanıcının şifresi, “password_verify” fonksiyonu ile şifreli olarak kontrol edilir.

Başarılı Giriş: Eğer giriş başarılıysa, kullanıcıya oturum açılır ve kullanıcı adı bir çerezde (cookie) saklanarak yönlendirme yapılır.

Hata Mesajı: Kullanıcı adı ya da şifre yanlışsa, hata mesajı kullanıcıya gösterilir.

login.php ve login.class.php Görsellerde

```

<?php
1 reference | 0 implementations
class LoginUser{
    3 references
    private $username;
    2 references
    private $password;
    3 references
    public $error;
    2 references
    public $success;
    1 reference
    private $storage = "../../datas/data.json";
    2 references
    private $stored_users;

    1 reference | 0 overrides
    public function __construct($username, $password){
        $this->username = $username;
        $this->password = $password;
        $this->stored_users = json_decode(json: file_get_contents(filename: $this->storage), associative: true);
        $this->login();
    }

    1 reference
    private function login(): string{
        foreach ($this->stored_users as $user) {
            if($user['username'] == $this->username){
                if(password_verify(password: $this->password, hash: $user['password'])){
                    setcookie(name: "kullaniciAdi", value: $this->username, expires_or_options: time() + 7200, path: "/");
                    header(header: "location: ../../index.html"); exit();
                }
            }
        }
        return $this->error = "Yanlış kullanıcı adı veya şifre";
    }
}

```

```

<?php require("login.class.php") ?>
<?php
✓ if(isset($_POST['submit'])){
    $user = new LoginUser(username: $_POST['username'], password: $_POST['password']);
}
?>
<!DOCTYPE html>
✓ <html lang="en">
✓ <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="style.css">
    <title>Log in form</title>
    <link rel="icon" href="../../images/logo2.png">
</head>
✓ <body>
✓ <div class="form-container">
    <!-- Logo -->
    <a href="../../index.html"></a>

    <form action="" method="post" enctype="multipart/form-data" autocomplete="off">
        <h2>Giriş Yap</h2>

        <input type="text" name="username" placeholder="Kullanıcı Adı">

        <input type="password" name="password" placeholder="Şifre">

        <button type="submit" name="submit">Giriş Yap</button>

        <div>Henüz bir hesabın yok mu? <a href="register.php">Kayıt ol</a></div>
        <p class="error"><?php echo @$user->error ?></p>
        <p class="success"><?php echo @$user->success ?></p>
    </form>
</div>
</body>
</html>

```

Eğer giriş yaparsanız da phpde oluşturduğumuz cookieyi login.js dosyasında alıp basit bir şekilde kullanabiliyoruz

```
window.onload = function() {
    function getCookie(name) {
        let nameEQ = name + "=";
        let ca = document.cookie.split(';');
        for (let i = 0; i < ca.length; i++) {
            let c = ca[i].trim();
            if (c.indexOf(nameEQ) === 0) {
                return c.substring(nameEQ.length, c.length);
            }
        }
        return null;
    }

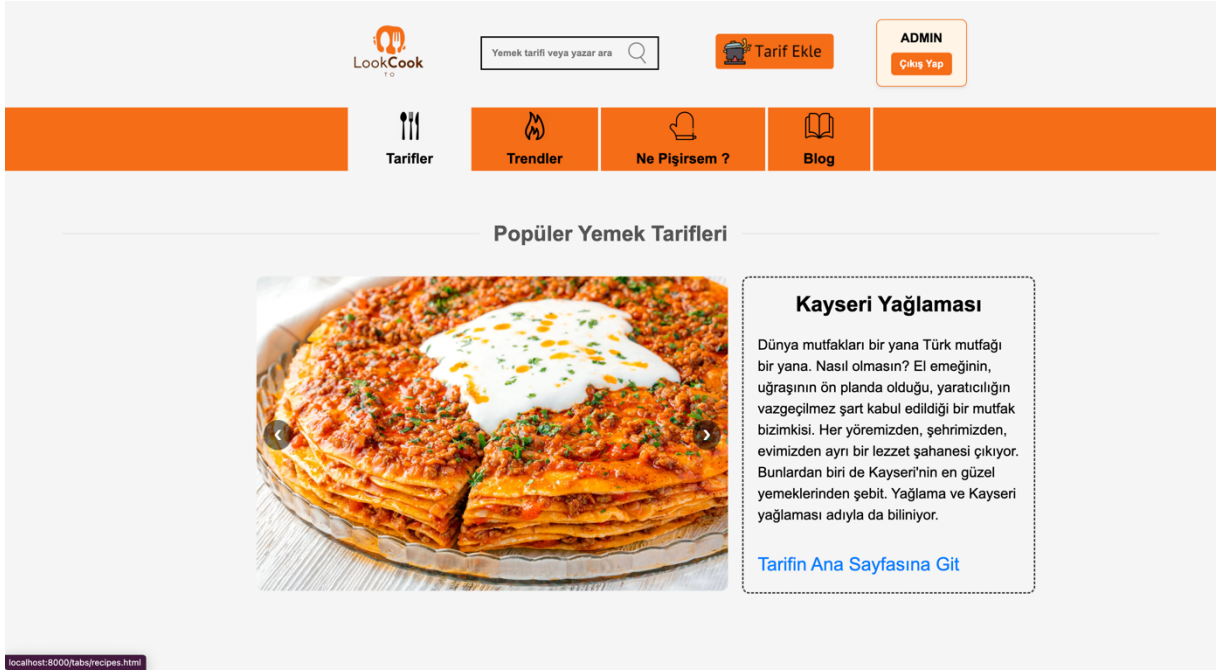
    var kullanıcıAdi = getCookie("kullanıcıAdi");
    const mainDiv = document.getElementById("user");
    if (kullanıcıAdi) {
        mainDiv.setAttribute("class", "user");
        mainDiv.innerHTML = `
        <p class="username"><b>${kullanıcıAdi.toUpperCase()}</b></p>
        <button class="logout-btn" onclick="deleteCookie('kullanıcıAdi')">Çıkış Yap</button>
        `;
    }
    else{
        mainDiv.setAttribute("class", "user0");

        document.getElementById("user").innerHTML = `
        <a href="tabs/login/login.php">
        
        </a>
        `;
    }
}

function deleteCookie(cookieName) {
    document.cookie = `${cookieName}=; expires=Thu, 01 Jan 1970 00:00:00 UTC; path=/;`;
    alert("Başarıyla çıkış yapıldı");
    window.location.href = "index.html";
}
```

Sitemizin sağ üst kısmında da giriş yapanın ismi ve altında da çıkış yap tuşu var eğer tuşa basarsa cooki silinip ekrana deleteCookie

fonksiyonu sayesinde “Başarıyla çıkış yapıldı” yazılıyor



Arama Barı

URL Parametresi:

window.location.search ile URL'deki parametreler alınır. Kullanıcı, arama terimini URL üzerinden gönderir. Bu parametre q olarak alınır ve küçük harfe dönüştürülür.

const query = params.get('q').toLowerCase(); ile arama sorgusu (q) alınır.

JSON Veri Yükleme:

fetch('../datas/en_yeniler.json') ile yemekler verisini içeren JSON dosyası sunucudan alınır.

.then(response => response.json()) ile JSON verisi çözümlenir ve yemekler değişkenine atanır.

Yemeklerin Filtrelenmesi:

Filtreleme Kriterleri: Yemekler, arama sorgusuyla karşılaştırılarak filtrelenir. Bu filtreleme şu kriterlere dayanır:

Başlık (title): Yemek başlığı, arama sorgusunu içeriyor mu kontrol edilir.

Kategori (category): Yemek kategorisi, arama sorgusunu içeriyor mu kontrol edilir.

Yazar (author): Yazar ismi, arama sorgusunu içeriyor mu kontrol edilir.

Filtreleme işlemi için yemekler.filter() fonksiyonu kullanılır.

Sonuçların Görüntülenmesi:

Eğer filtrelenmiş yemeklerin sayısı sıfır ise, kullanıcıya "Sonuç bulunamadı." mesajı gösterilir.

Eğer eşleşen yemekler varsa, her bir yemek için HTML kartı (card) dinamik olarak oluşturulur ve sonucListesi adlı HTML ögesinin içine eklenir.

Yemekler hakkında:

Görsel (imgSrc),

Kategori (category),

Başlık (title),

Yazar (author),

Hazırlanma süresi (time),

Puan (rating) gibi bilgileri içeren bir kart tasarımı oluşturulur.

Hata Yönetimi:

Eğer JSON dosyasının yüklenmesinde bir hata oluşursa, catch bloğu devreye girer ve konsola hata mesajı yazdırılır.

```
const params = new URLSearchParams(window.location.search);
const query = params.get('q').toLowerCase();

fetch('../datas/en_yeniler.json')
  .then(response => response.json())
  .then(yemekler => {
    const sonucListesi = document.getElementById('sonuclar');

    const filtrelenmisYemekler = yemekler.filter(yemek =>
      yemek.title.toLowerCase().includes(query) ||
      yemek.category.toLowerCase().includes(query) ||
      yemek.author.toLowerCase().includes(query)
    );

    if (filtrelenmisYemekler.length === 0) {
      sonucListesi.innerHTML = "<p>Sonuç bulunamadı.</p>";
    } else {
      filtrelenmisYemekler.forEach(item => {
        let card = `
          <div class="card">
            
            <div class="card-info">
              <div class="card-badge">${item.badge}</div>
              <a href="/tabs/recipes/tarifler.html?category=${item.category}">
                <div class="card-category">${item.category}</div>
              </a>
              <a href="recipes/${item.link}">
                <h3 class="card-title">${item.title}</h3>
              </a>
              <p class="card-author">${item.author}</p>
              <div class="card-details">
                <span class="card-time">${item.time}</span>
                <span class="card-rating">${item.rating}</span>
              </div>
            </div>
          </div>
        `;
        sonucListesi.innerHTML += card;
      });
    }
  })
  .catch(error => console.error("JSON yüklenirken hata oluştu:", error));
```

Bu da ara.html dosyasının script kısmında yer alan kodumun görseli.

Tarif Ekleme

```
<script>
  document.getElementById("tarifForm").addEventListener("submit", function(event) {
    event.preventDefault();

    const yeniTarif = {
      title: document.getElementById("title").value,
      imgSrc: document.getElementById("imgSrc").value,
      imgAlt: document.getElementById("imgAlt").value,
      time: document.getElementById("time").value + "dk",
      category: document.getElementById("category").value,
      author: document.getElementById("author").value,
      ingredients: document.getElementById("ingredients").value,
      instructions: document.getElementById("instructions").value,
      rating: "*****",
      badge: "YENİ",
      link: "#"
    };

    console.log("Yeni Tarif:", JSON.stringify(yeniTarif, null, 2));

    alert("Tarif başarıyla eklendi! (Şu an sadece console'a yazılıyor.)");
  });
</script>
```

Event Listener (Olay Dinleyicisi):

```
document.getElementById("tarifForm").addEventListener("submit", function(event) {...});
```

Burada, tarifForm id'sine sahip form elemanına bir "submit" (gönderme) olayı ekleniyor. Bu, form gönderildiğinde çalışacak olan işlevi belirler.

Form Gönderimini Engelleme:

```
event.preventDefault();
```

Formun sayfayı yeniden yüklemesini (varsayılan form gönderim davranışını) engelleriz. Yani, form verileri gönderildikten sonra sayfa yenilenmez.

Form Verilerinin Alınması:

Yeni tarif için gerekli olan tüm veriler, form elemanlarından alınır:

title: Tarifin adı.

imgSrc: Resmin kaynağı (URL).

imgAlt: Resmin alternatif metni.

time: Tarifin hazırlanma süresi.

category: Tarifin kategorisi.

author: Tarifi yazan kişi.

ingredients: Tarifin malzemeleri.

instructions: Tarifin hazırlanış talimatları.

rating: Tarife verilen puan (bu örnekte sabit olarak ★★★★★).

badge: Tarife etiket (bu örnekte YENİ olarak belirlenmiş).

link: Tarifin bağlantısı (bu örnekte # olarak bırakılmış).

Yeni Tarifin Konsola Yazdırılması:

```
console.log("Yeni Tarif:", JSON.stringify(yeniTarif, null, 2));
```

Formdan alınan verilerle bir yeniTarif nesnesi oluşturulur. Bu nesne, JSON formatına çevrilir ve konsola yazdırılır.

JSON.stringify(yeniTarif, null, 2) kullanılarak nesne okunabilir bir biçime dönüştürülür.

Kullanıcıya Uyarı Gösterme:

```
alert("Tarif başarıyla eklendi! (Şu an sadece console'a yazılıyor.);
```

Kullanıcıya, tarifin başarıyla eklendiğine dair bir uyarı gösterilir. Bu uyarı, tarifin sadece konsola yazdırıldığını belirtir.