1. **Tablolarda id gorunmemeli.**

   Sadece view tarafinda degisiklik yaparak cozuldu.

2. **Url kisminda idler gorunmemeli. Bunun icin modal yapisi kullanilabilir.**

   Controller ve view tarafinda yapilan degisiklikler ile cozuldu. Eski edit.cshtml ve delete.cshtml dosyalari kaldirildi. Yerlerine _EditModalPartial.cshtml ve _SetPasswordModalPartial.cshtml geldi. Ayrica Controller tarafinda edit ve delete methodlari EditModal ve DeleteModal ile degistirildi.

3. **Aktif pasif durumu toggle değil.**

   Toggle butonları klasik form submit **yerine AJAX (fetch)** ile çalışacak şekilde revize edildi. Controller metodları **JSON dönecek** biçimde düzenlendi ve başarılı işlem sonrası ilgili satırın aktif durumu ile buton metni dinamik olarak güncellendi. Böylece sayfa yenileme kaldırılarak kullanıcı deneyimi iyileştirildi.

4. **Iskontoda klavye ile negative deger girilebiliyor. Ornegin -10 yuazip iskonto uygulanmaya calisinca indirim yapilmiyor.**

   Views/Siparisler/Take.cshtml içinde iskonto input'u, tarayıcıların type="number" alanlarında - / e / E gibi karakterlere izin verebilmesi nedeniyle daha kontrollü bir yapıya alındı ve **yalnızca sayısal girişe** uygun hale getirildi.

   wwwroot/js/siparis-take.js tarafında iskonto alanına anlık sanitize (input sırasında temizleme) ve 0–100 aralığına **clamp** uygulanarak kullanıcı klavye ile negatif veya sayı dışı karakter giremez duruma getirildi.

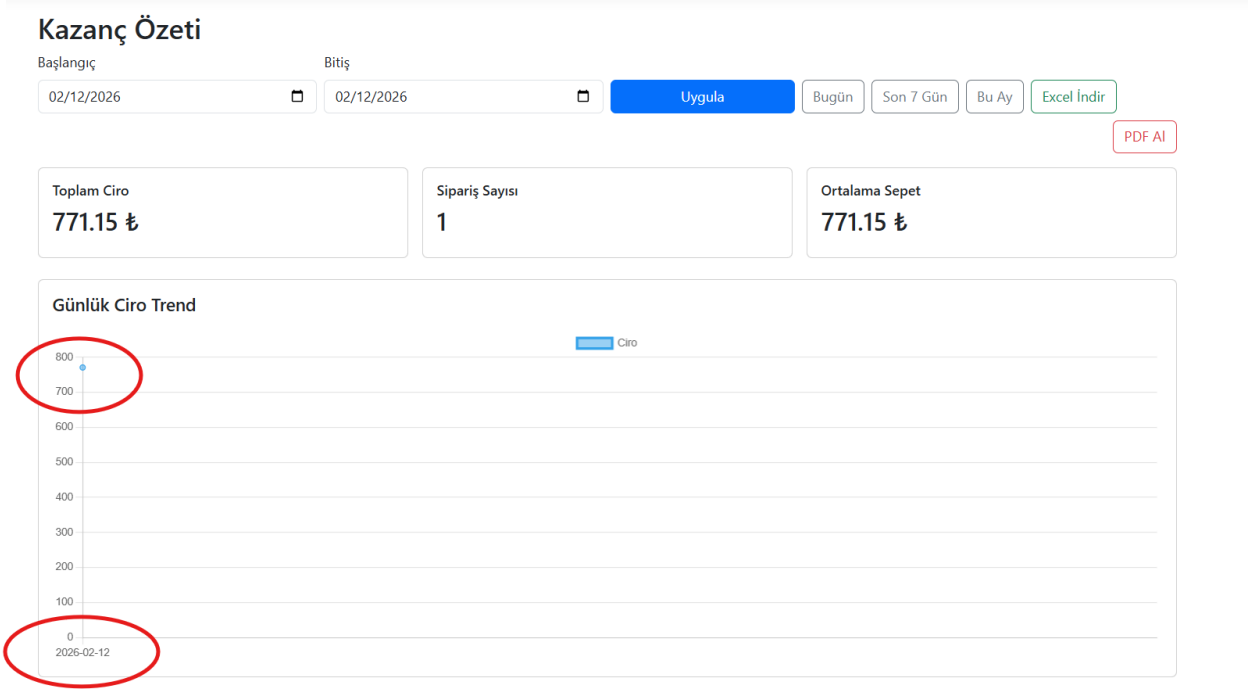
   SiparislerController.UpdateDiscount içinde SiparisId ve IskontoOran için kontrol eklenerek, istemci tarafı atlanmış olsa bile IskontoOran < 0 veya > 100 durumlarında istek **BadRequest** ile reddedilecek şekilde garanti altına alındı.

   SiparisService.UpdateDiscountRate içinde de aynı aralık doğrulaması tekrarlanarak iş kuralı **servis katmanında** güvenceye alındı (controller bypass edilse dahi geçersiz iskonto uygulanamaz).

5. **Admin kendi hesabinin rollerinden admini kaldiramamali.**

PersonelService.Update metoduna, kullanıcının kendi kaydını güncellerken mevcut Admin rolünü kaldırmasını engelleyen iş kuralı eklendi. Böylece **server-side** güvence sağlandı. Ayrıca _EditModalPartial.cshtml içinde kendi hesabını düzenlerken Admin checkbox'ı **disable** edilerek UI tarafında da görsel kısıt uygulandı.

6. **Rapor kisminda bugun kismi gun bitmedigi icin mi gelmiyor?**



Geliyor. Sadece tabloda x ve y degeri olarak 1 er deger oldugu icin grafik olusmuyor. Tek bir nokta olusuyor. Bu da tabloda veri yansimadi yanilgisi olusturabiliyor.

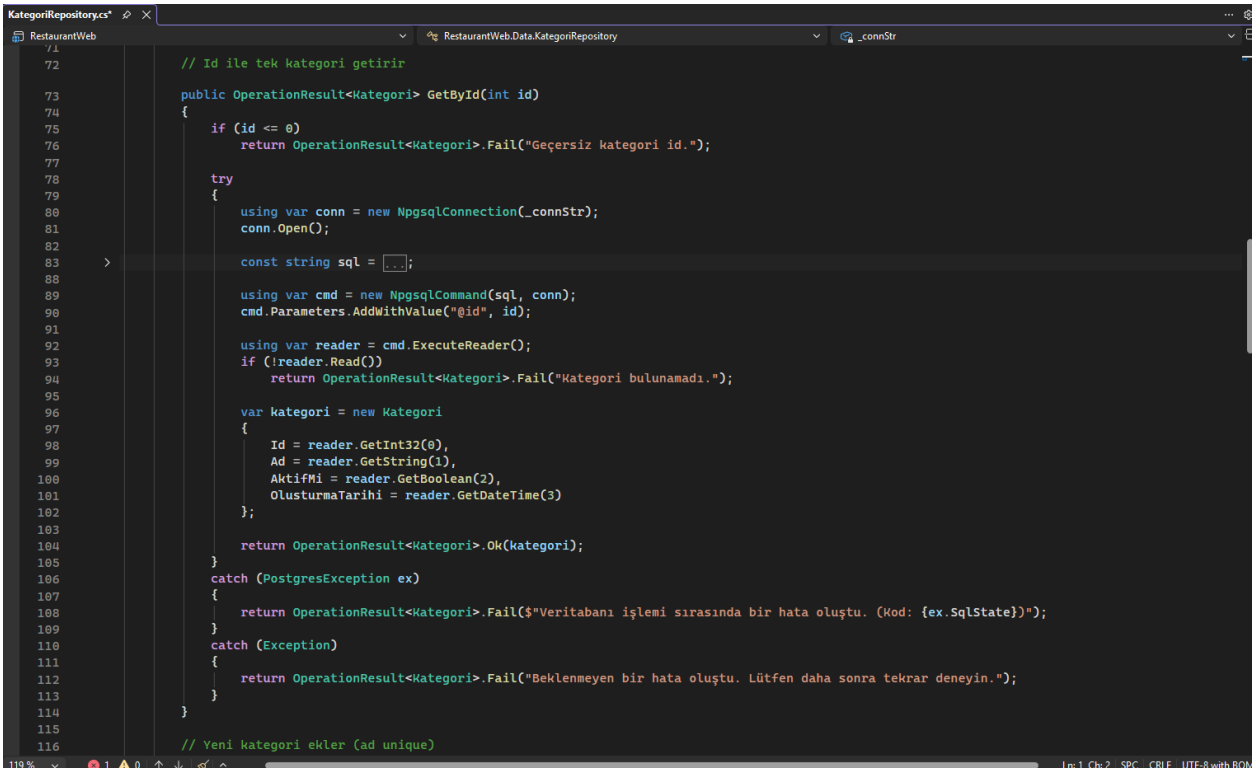
7. **Sepetteki ürünleri adisyona eklemeden sonra ve siparişi kapatmadan sonra başarı mesajı dönmeli.**

Siparisler/Take ekranında **Kaydet (Submit)** işlemi başarılı olduğunda, siparis-take.js içinde sunucudan dönen message kullanılarak sayfa yenilenmeden Bootstrap toast ile "Sepetteki ürünler adisyona eklendi" bilgilendirmesi gösterilecek şekilde güncellendi.

Ödeme Al / Siparişi Kapat (Close) işleminde ise controller tarafında TempData["Success"] set edilerek kullanıcı Masalar/Board ekranına yönlendirildiğinde sağ üstte “Ödeme başarıyla alındı” toast mesajının görünmesi sağlandı.

8. Kategori Yönetimi kısmında Try-Catch yok

Sadece kategori değil herhangi bi bölümün Controller tarafında özellikle herhangi bi hata yakalama işlemi yapılmadı. Hata yakalama ya service katmanına ya da repository katmanına bırakıldı. (basit hatalar if ile yakalandı.) Burada temel amaç, örneğin database kaynaklı bir hatanın Controller katmanında tespit edilmesinin imkanı olmaması ve de controller kısmını try catch ile uzatılarak kod okunurlugunun düşmesini önlemek.



9. Session Management (giriş yapan kullanıcı bilgisi)

Mevcut projede bulunanlar:

ASP.NET Core Cookie Authentication

- Login olunca cookie oluşturuluyor
- Sliding expiration var
- 8 saat expire var
- Logout var

Teknik olarak basit seviyede session management var.

Fakat geliştirilebilir.

Mevcut Cookie ayarları:

```
// Cookie tabanlı kimlik doğrulama
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Auth/Login";
        options.LogoutPath = "/Auth/Logout";
        options.AccessDeniedPath = "/Auth/Denied";
        options.Cookie.Name = "RestaurantWeb.Auth";
        options.SlidingExpiration = true;
        options.ExpireTimeSpan = TimeSpan.FromHours(8);
    });
```

Yeni hali:

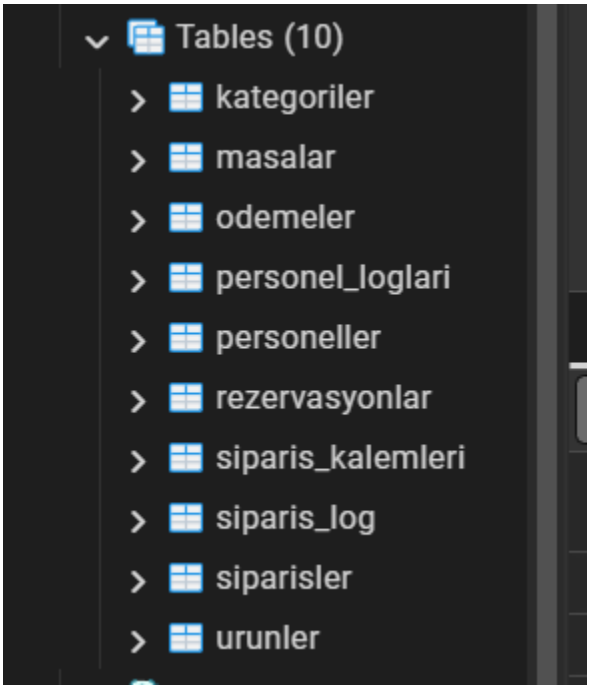
```
// Cookie tabanlı kimlik doğrulama
builder.Services.AddAuthentication(CookieAuthenticationDefaults.AuthenticationScheme)
    .AddCookie(options =>
    {
        options.LoginPath = "/Auth/Login";
        options.LogoutPath = "/Auth/Logout";
        options.AccessDeniedPath = "/Auth/Denied";
        options.Cookie.Name = "RestaurantWeb.Auth";

        options.Cookie.HttpOnly = true;
        options.Cookie.SecurePolicy = CookieSecurePolicy.Always;
        options.Cookie.SameSite = SameSiteMode.Lax;

        options.SlidingExpiration = true;
        options.ExpireTimeSpan = TimeSpan.FromHours(8);

        options.Events = new CookieAuthenticationEvents
        {
            OnValidatePrincipal = async ctx =>
            {
                var userIdStr = ctx.Principal?.FindFirstValue(ClaimTypes.NameIdentifier);
                if (!int.TryParse(userIdStr, out var userId))
                {
                    ctx.RejectPrincipal();
                    await ctx.HttpContext.SignOutAsync();
                    return;
                }

                using var scope = ctx.HttpContext.RequestServices.CreateScope();
                var repo = scope.ServiceProvider.GetRequiredService<PersonelRepository>();

                var activeRes = repo.IsActiveById(userId);

                // bulunamadı / hata / pasif -> oturumu düşür
                if (!activeRes.Success || activeRes.Data == false)
                {
                    ctx.RejectPrincipal();
                    await ctx.HttpContext.SignOutAsync();
                }
            }
        };
    });
```

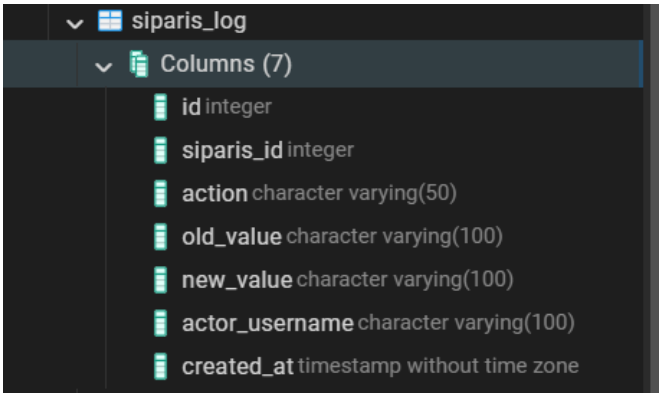
Bu değişiklikler sonrası:

- Kullanıcı pasif yapılırsa aktif oturum düşüyor.
- Admin bir kullanıcıyı pasif edince o anki session invalidate oluyor.

10. Sipariş log tablosu ve loglama

DB'de Sipariş Log tablosu bulunuyor:





Loglama işlemi ise, kullanıcı siparişte bir değişiklik yaptığında siparişi açtığında ya da kapattığında ya da iskonto tutarını değiştirdiğinde gerçekleşiyor.

Kullanıcılar, siparişler sekmesindeki sipariş geçmişinden ilgili siparişin detay kısmına girdiğinde sayfanın alt kısmında İşlem Geçmişini yani SiparisLog unu bulabilir.
Burada işlemin tarihi, işlemin türü, işlem öncesi durum, sonrası durum ve de tüm bu işlemlerin kim tarafından yapıldığı bilgisi tutulmaktadır.