**UYGULAMALI MÜHENDİSLİK EĞİTİMİ HAFTALIK RAPOR**

**Öğrenci No:** 210290044

**Adı Soyadı:** Yusuf Bozkurt

**Bölümü:** Yazılım Mühendisliği

**İşyeri Bilgisi:** Haberler.com

**Rapor No:** 2

**Rapor Tarih Aralığı:** 03.03.2025 – 07.03.2025

**Rapor Gönderme Tarihi:**09.03.2025

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hafta  1 | Hafta  2 | Hafta  3 | Hafta  4 | Hafta  5 | Hafta  6 | Hafta  7 | Hafta  8 | Hafta  9 | Hafta  10 | Hafta  11 | Hafta  12 | Hafta  13 | Hafta  14 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |

İÇİNDEKİLER

**KONU**  **SAYFA**

**GÜN1:** Hata taraması ve Kimlik doğrulama projesi

(Usercontroller,Rolecontroller oluşturuşması)...............................................................................3-4

**GÜN2:** Hata testi ve Kimlik doğrulama projesi devam.................................................................5-6

**GÜN3:** Hata testi ve Test otomasyonu(navbar ve footer için).......................................................7-8

**GÜN4:** Test Otomasyonu(devam) ve hataların mail olarak gönderilmesi....................................9-10

**GÜN5:** Hata taraması ve Task Scheduler......................................................................................11-12

**KAYNAKÇA:**................................................................................................................................13

**ŞEKİL VE TABLOLAR:**

**ŞEKİL** **SAYFA**

**Şekil 1.1**........................................................................................................................................3

**Şekil 1.2**........................................................................................................................................3

**Şekil 1.3**........................................................................................................................................4

**Şekil 2.1**........................................................................................................................................5

**Şekil 2.2**........................................................................................................................................5

**Şekil 2.3**........................................................................................................................................6

**Şekil 2.4**........................................................................................................................................6

**Şekil 3.1**........................................................................................................................................7

**Şekil 3.2**........................................................................................................................................7

**Şekil 3.3**........................................................................................................................................8

**Şekil 3.4**........................................................................................................................................8

**Şekil 3.5**........................................................................................................................................8

**Şekil 4.1**........................................................................................................................................9

**Şekil 4.2**........................................................................................................................................9

**Şekil 4.3**........................................................................................................................................10

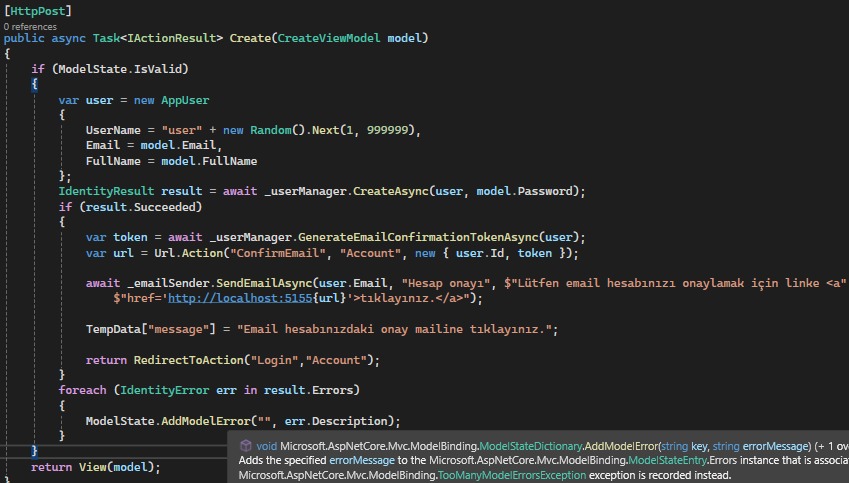
**Şekil 4.4**........................................................................................................................................10

**Şekil 5.1**........................................................................................................................................11

**Şekil 5.2**........................................................................................................................................12

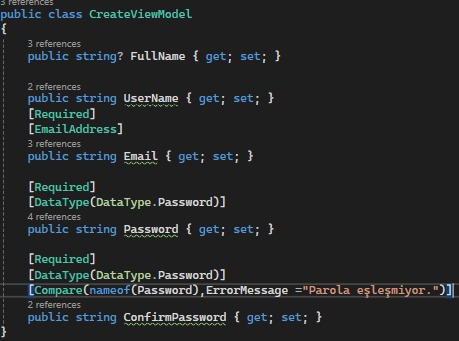
**1.GÜN**

Bugün yine rutin işim olan haberler.com ve sondakika.com da hata tespiti için siteyi test etmeye başladım bugün sitede herhangi bir bug bulamadım, bu işlem günlük olarak yaklaşık 2 saatimi almakta, Bu işin ardından mentörüm udemy den sadık turanın .net kursunu incelememi istedi ve bende bu kursun içeriğinde bulunan bir kimlik doğrulama projesini yapmaya başladım. Bu projede edindiğim amaç, kullanıcı oluştukran sonra e mail hesabına bir mail alması ve bu mailden hesap doğrulaması yaptıktan sonra hesaba erişim sağlayabilmesi. Öncelikle UsersController ve RoleController adında 2 adet controller oluşturdum ve Controllerların arayüzü için Views klasörü altında Role ve Users adında iki adet dosya oluşturdum. İlgili Controllerların action larını ilgili klasörülerinden altında html kullanarak arayüz olarak oluşturacağım. İlk olarak UsersControllerın altında create actionu oluşturdum. Bu action kullanıcı oluşturmaya yarayacak.

****

**Şekil 1.1**

Kullanıcı oluşturma işlemi yapıldıktan sonra email gönderen bir service yazacağım bu kodda görmüş olduğunuz SendEmailAsync metodu şuan işlevsiz. Ardından Models klasörü altında CreateViewModel adında bir model oluşturdum, Bu modelin içeriğinde Fullname, UserName, Email, Password, ConfirmPassword alanları yer alıyor.

****

**Şekil 1.2**

Ardından Views-Users altında html ile yazdığım, Fullname, UserName, Email, Password, ConfirmPassword alanları yer alan bir arayüz oluşturdum. Ve bu actionun testini yaptım. RoleController u yazmaya başladım bu kısımda rol ekliyebiliyoruz ve kullanıcılara çeşitli roller atayabiliyoruz.

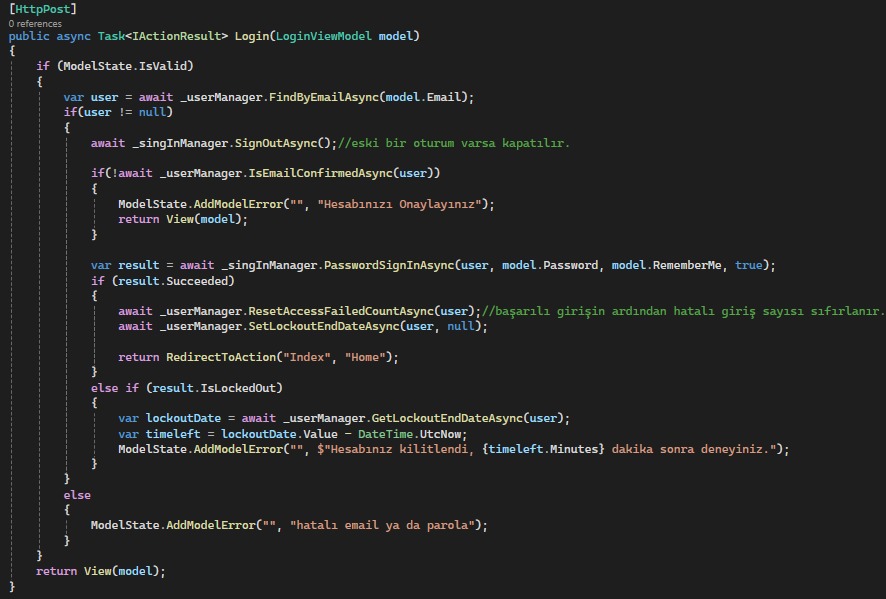
****

**Şekil 1.3**

Rol atama işlemini ayrı bir sayfaya taşıdım. AssignRole metodu, mevcut rolleri ViewBag.Roles ile sayfaya gönderiyor ve kullanıcıdan seçim yapmasını sağlıyor. Seçilen rol AssignRoleToUser metoduna gönderildiğinde, önce rolün geçerli olup olmadığı kontrol ediliyor. Eğer geçerliyse, kullanıcıya ekleniyor ve Index sayfasına yönlendiriliyor. Böylece işlemi daha düzenli ve yönetilebilir hale getirdim. Role Atama ile ilgili arayüzü ise Views-Role klasörü altında oluşturdum.

**2.GÜN**

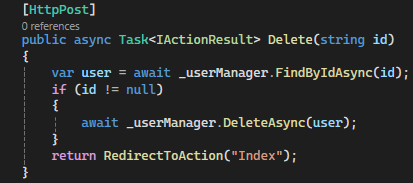
Bugün sabah yine hata taraması yaptım herhangi bir problem yoktu sitede, Daha sonra projeme devam etmeye başladım. AccountController adında bir controller kurup içine kullanıcı girişi için login actionu ekledim.

****

**Şekil 2.1**

Kullanıcının giriş yapmasını sağlayacak bir metot oluşturdum. Öncelikle, girilen verilerin doğruluğunu kontrol ediyorum. Eğer email adresiyle eşleşen bir kullanıcı varsa, önce varsa açık oturumu kapatıyorum. Daha sonra, kullanıcının email adresinin onaylanıp onaylanmadığını kontrol ediyorum. Eğer email onaylanmamışsa, hata mesajı gösteriyorum. Kullanıcı bilgileri doğruysa giriş işlemi gerçekleşiyor ve başarısız giriş sayısı sıfırlanıyor. Ancak, eğer hesap kilitlenmişse, ne kadar süre kilitli olduğunu hesaplayıp kullanıcıya bilgilendirici bir hata mesajı veriyorum. Şifre veya email yanlışsa da hata mesajı gösteriyorum. Başarılı giriş durumunda ana sayfaya yönlendiriyorum, aksi halde sayfayı hata mesajlarıyla tekrar yüklüyorum. Bu login actionun ardından Views-Account dosyası oluşturup login in arayüz tasarımını buraya kodladım.

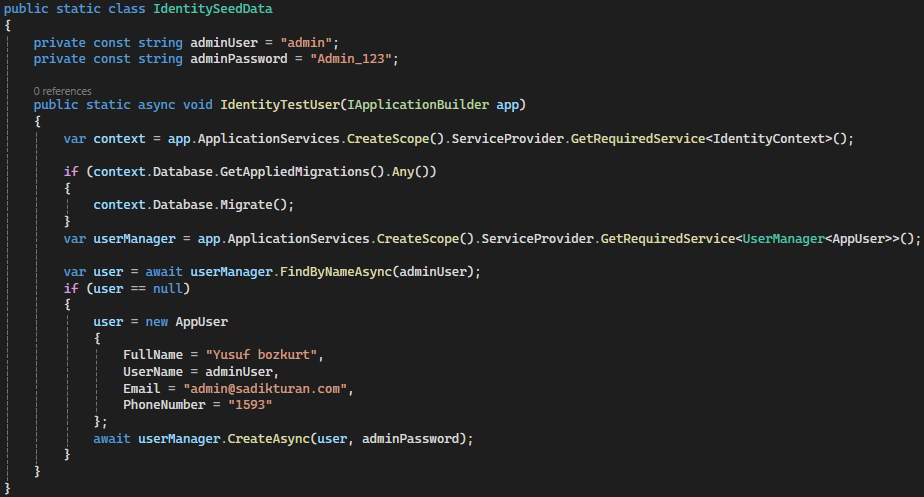
UsersController da yer alan Create metodunu AccountController a taşıdım ve users da sadece admin tarafından yapılabilen Güncelleme listeleme ve silme metodları yazdım.



Şekil 2.2

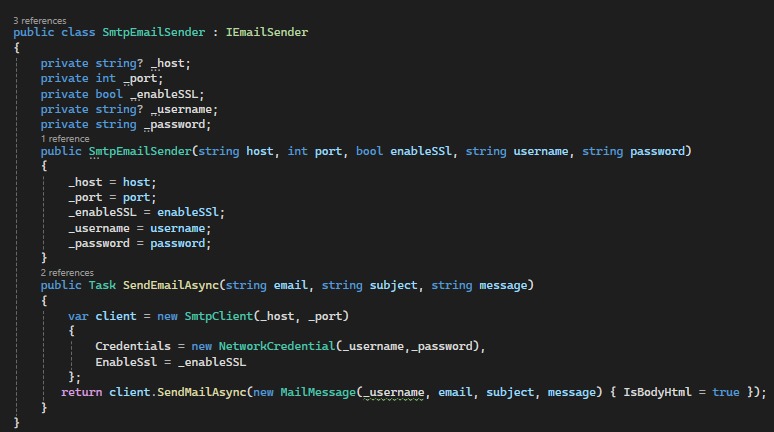
Şekil 2.2 de UsersController daki sadece adminin yönetebildiği bir silme metodunu paylaştım.

Ardından sistemde varsayılan bir admin kullanıcısı oluşturmak için IdentitySeedData adında bir dosya oluşturdum.



**Şekil 2.3**

Uygulama başlatıldığında varsayılan bir admin kullanıcısı eklemek için bu kodu yazdım. Öncelikle veritabanındaki eksik migrasyonları uyguluyorum. Daha sonra UserManager ile sistemde "admin" adlı bir kullanıcı olup olmadığını kontrol ediyorum. Eğer yoksa, kullanıcı adı, e-postası ve şifresiyle yeni bir admin hesabı oluşturuyorum. Böylece her zaman en az bir admin kullanıcısı olduğundan emin oluyorum.

****

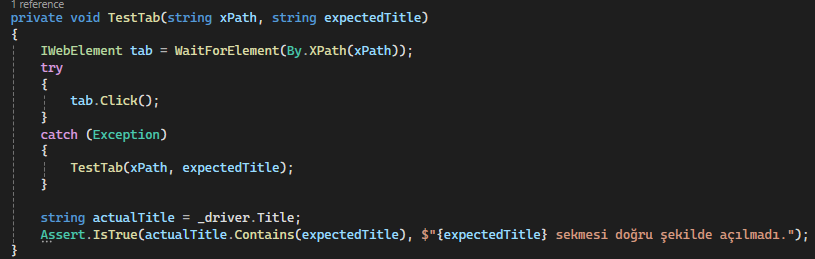
**Şekil 2.4**

SMTP ile e-posta göndermek için bir SmtpEmailSender sınıfı yazdım. Bu sınıf, belirttiğim SMTP sunucusu üzerinden kimlik doğrulaması yaparak e-posta göndermemi sağlıyor. İlk olarak, kurucu metodunda SMTP sunucu adresi, port numarası, SSL kullanımı, kullanıcı adı ve şifre gibi bilgileri alıp saklıyorum. Daha sonra SendEmailAsync metodunu kullanarak bir SmtpClient nesnesi oluşturuyorum ve bağlantıyı sağlıyorum. Burada kullanıcı adı ve şifre ile kimlik doğrulaması yapılıyor, ardından e-posta bilgilerini içeren bir MailMessage nesnesi oluşturuyorum. Gönderici ve alıcı bilgilerini ekleyip, mesajın html formatında olmasını sağladıktan sonra SendMailAsync metodunu çağırarak e-postayı gönderiyorum.

**3.GÜN**

Sabah yine haberler.com ve sondakika.com un hata taramasını gerçekleştirdikten sonra mentörüm yeni görev verdi, ben bu testleri hep manuel olarak elimle gerçekleştiriyordum bunun otomasyonunu yazmamı istedi benden. Sitede birçok sayfa olduğu için sadece belirli bir kısım için bir test otomasyonu yazacaktım tabi, ilk olarak navbar kısmındaki sekmeleri test etmeye karar verdim. Daha önce hiç test otomasyonu yazmadığım için baya bir video izledim Daha sonra selenium ile testleri gerçekleştirmeye karar verdim.

**Selenium;** web uygulamalarını test etmek için kullandığım bir otomasyon aracı. Web sayfalarında yaptığım işlemleri otomatikleştiriyor ve test ediyor. Tarayıcılarla etkileşimde nasıl çalışıyorsam, Selenium da aynı şekilde bu etkileşimleri simüle ediyor ve uygulamanın doğru şekilde çalışıp çalışmadığını kontrol ediyor. Böylece manuel testleri hızlandırıyor ve hataları daha çabuk bulmama yardımcı oluyor. Projeme başlamak için önelikle selenium.webdriver ve selenium.support paketlerini indirdim ve ardından navbar için ilk kodumu yazmaya başladım.



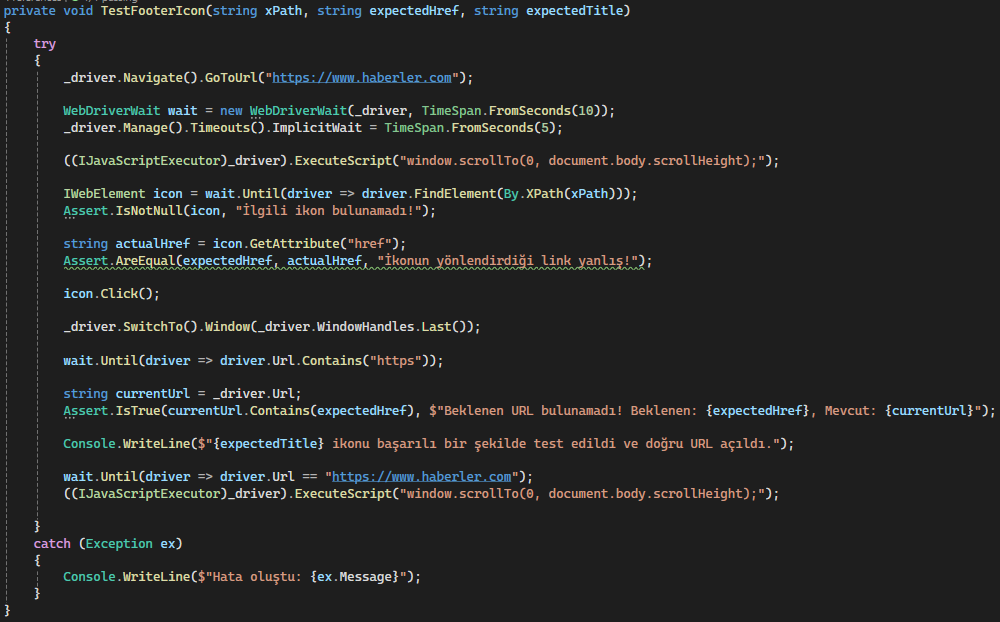
**Şekil 3.1**

Şekil 3.1 de ki metod, verdiğim xPath ile bir sekmeyi tıklamaya çalışıyor. Eğer sekme tıklanamazsa, varsa reklâmı kapatıp tekrar deniyor. Sonrasında sayfa başlığını alıp, beklediğim başlıkla karşılaştırarak doğru şekilde açılıp açılmadığını kontrol ediyorum.

****

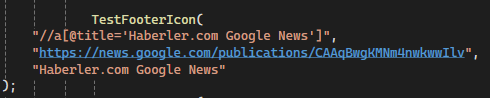
**Şekil 3.2**

Yazmış olduğum TestTab metodunun verilerini burada giriyorum. Önce "<https://www.haberler.com>" ana sayfasına gidiyorum, sonra farklı sekmeleri test etmek için TestTab metodunu kullanıyorum. Her bir sekmeyi, ilgili XPath ile tıklayıp başlıklarını kontrol ediyorum. En son olarak, "Kadın" sekmesini test edip, sayfada geri dönüyorum

****

**Şekil 3.3**

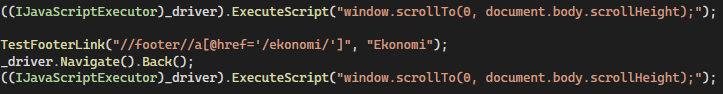
Bu metod, "<https://www.haberler.com>" sayfasına gidip, belirlediğim ikonu test ediyor. Önce sayfanın altına kaydırıp, ikonu buluyorum. Sonra tıklayıp, yeni açılan pencerede doğru URL'nin olup olmadığını kontrol ediyorum. Her şey doğruysa, başarılı bir test mesajı yazıyorum. Eğer bir hata olursa, hata mesajını konsola yazdırıyorum.



**Şekil 3.4**

Örneğin Şekil 3.4 de Google news butonunun işlevselliğini test etmek için bu butonun title ve href ini tanımlamışız. Footerdaki 8 adet sosyal medya ikonunu da test ettim fakat örnek olsun diye 1 tane bıraktım sadece.

Bir de footer daki linkler ve butonlar için bir metod yazdım, bu metod testab metodu gibi yolunnu ve title ini parametre olarak aldığı için yani her iki metod aynı olduğu için resmini paylaşmaya ihtiyaç duymadım. Fakat nasıl çağırdığım ile ilgili resim paylaşacağım alt kısımda.

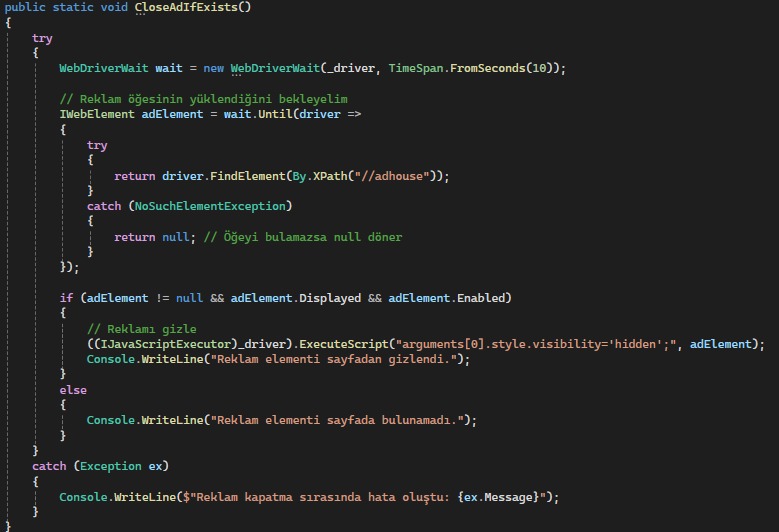


**Şekil 3.5**

Şekil 3.5 te yer alan kod bloğu sayfanın en altına kaydırarak "Ekonomi" linkine tıklıyor, sonra geri dönüp tekrar sayfanın altına kaydırıyor. Footerdaki yaklaşık 15 tane linke test sağladım sadece 1 tanesini örnek olarak paylaştım.

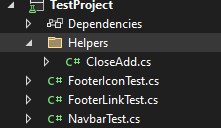
**4.GÜN**

Bu yazdığım test otomasyonunda sürekli reklamlara denk geliyordum ve bu reklamları kapatabilen bir metod yazdım.

****

**Şekil 4.1**

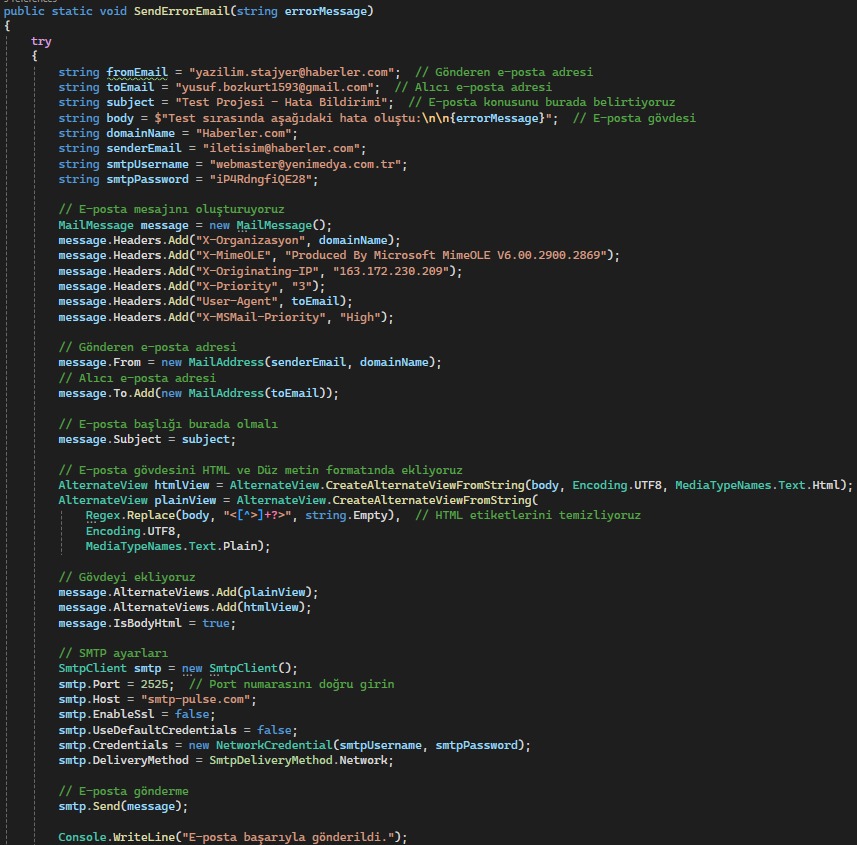
Bu metod reklam çıktığı zaman ekranda gizliyor.

****

**Şekil 4.2**

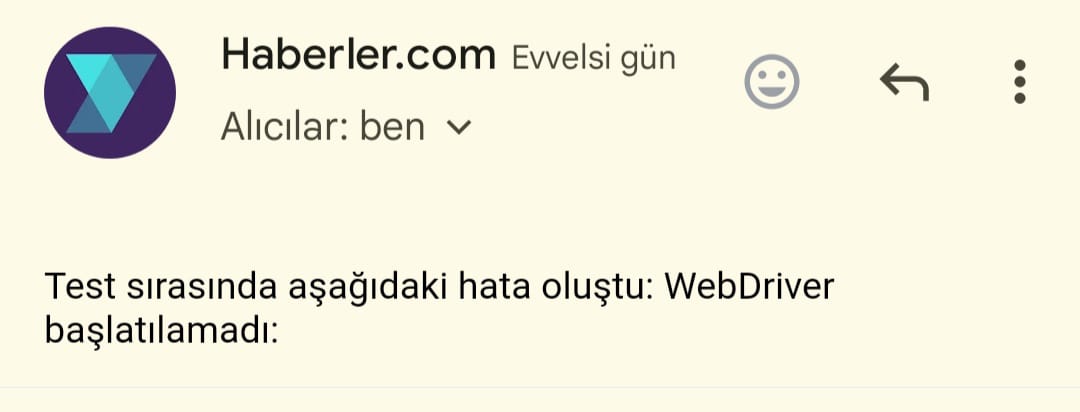
Bu işlemleri, metodları sadece 1 sayfa üzerinde tasarladığım için mentörüm dependcy injection kuralına bağlı kalarak projeyi tasarlamamı istedi, ben de Şekil 4.2 de olduğu gibi bir dosya yapısı oluşturdum ardından bağımlılıkları dosyaların içinden çağırdım.

Test otomasyonunun belli kısımları bittikten sonra mentörüm, test sırasında hata aldığımız zaman hatayı mail olarak bize iletmesini sağlamamı istedi.

****

**Şekil 4.3**

Bu metod, bir hata mesajını belirlediğim alıcıya e-posta olarak gönderiyor. Önce gönderen ve alıcı e-posta adreslerini, konu ve içeriği belirliyorum. E-posta, hem HTML hem de düz metin formatında hazırlanıyor. Sonra, SMTP ayarlarıyla sunucuya bağlanıp e-posta gönderiyorum. E-posta başlıklarını da ekliyorum ve hata mesajı başarıyla gönderildiğinde bunu konsola yazdırıyorum.



**Şekil 4.4**

Şekil 4.4 test sırasında aldığım bir hatanın mail olarak geldiği bir ekran görüntüsü var.

**5.GÜN**

haberler.com ve sondakika.com arayüzünde hata taraması yaptım ve haberler.com android web sitesi için şu hataları tespit ettim;

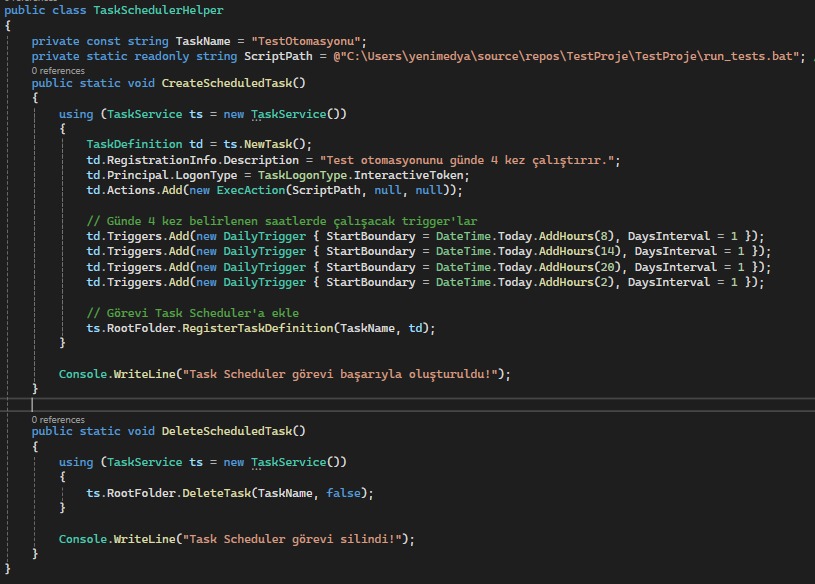
1-) https://www.haberler.com/abdullah-ocalan/ kısmındaki kayan resmin altındaki imlec ekranın ortasına geliyor, bütün sayfayı yataydan kaplamalı

2-)https://www.haberler.com linki yani anasayfa hariç diğer tüm sayfalarda kayan haberler kısmında sürükleyerek geçemiyouruz sadece tuşa basmak zorundayız.

3-) finans-borsa-teknik analiz butonuna basıldığında "sayfayı görüntülemeye yetkiniz yoktur uyarısı alınıyor."

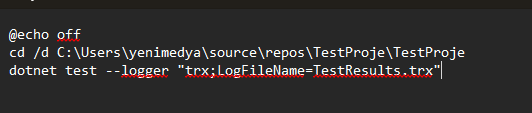
4-)https://www.haberler.com/yazarlar/ kısmındaki yazar ol butonu aktif değil

Bu hataları raporlayıp ilgili mühendise ilettim, ardından yazdığım test otomasyonu için **Task Scheduler** konusunu araştırmaya başladım. Task Scheduler, görev zamanlayıcısı anlamına gelir Microsoft Windows'da bilgisayar programlarını veya betiklerini önceden tanımlanmış zamanlarda veya belirtilen zaman aralıklarından sonra başlatan bir iş zamanlayıcısıdır. Öncelikle bu oluşturduğum Test otomasyonu için, windows ta bulunan görev zamanlayıcısı kısmından otomatik olarak bir task scheduler kurdum, günde 4 defa 08.00, 14.00, 20.00, 02.00 saatlerinde çalışan bir task scheduler ayarladım. Ardından mentörüm .net ile bir Task Scheduler kurmamı istedi ve bende izldeiğim videolar ve yapay zekadan yararlanarak bir task scheduler yazmaya çalıştım bu da yine otomatik oluşturduğum kısımdaki gibi günde 4 defa ve belirilen saatlerde çalışacak, bu kısımda yazdığım kod şu şekilde;



Şekil 1.1

Bu sınıfı, test otomasyonumu belirli saatlerde otomatik olarak çalıştırmak için yazdım. CreateScheduledTask metodu, Windows Task Scheduler’a bir görev ekleyerek **run\_tests.bat** dosyasını günde dört kez çalıştırıyor. Eğer bu görevi kaldırmak istersem, DeleteScheduledTask metodunu kullanarak Task Scheduler’dan siliyorum. Run\_tests.bat dosyasının içeriği;



Şekil 1.2

Bu bat dosyasını, testlerimi otomatik olarak çalıştırmak için hazırladım. Önce test projemin olduğu dizine gidiyor, ardından dotnet test komutunu çalıştırarak testleri koşturuyor ve sonuçları **TestResults.trx** dosyasına kaydediyor.

Bugünkü işlerim bu kadardı.

**KAYNAKÇA:**

**[1]** https://www.techtarget.com/searchsecurity/definition/authentication

**[2]** https://www.udemy.com

**[3]** https://medium.com/@humeyrakoseogluu/test-otomasyonu-yaz%C4%B1l%C4%B1m-geli%C5%9Ftirmenin-h%C4%B1zl%C4%B1-ve-g%C3%BCvenilir-yolu-36fce82dff9e

**[4]** https://www.qodo.ai/blog/how-to-write-test-cases-with-automation-tools/

**[5]** Hartenstein, A., & Kaur, G. (2021). Mastering Selenium WebDriver. Packt Publishing, Birmingham, UK.

**[6]** Kurniawan, A., & Tan, S. (2020). "Automated Web Testing with Selenium: Challenges and Solutions," Journal of Software Testing, vol. 45, no. 4, pp. 1123-1136, October, 2020.

**[7]** https://www.windowscentral.com/how-create-automated-task-using-task-scheduler-windows-10

**[8]** https://learn.microsoft.com/en-us/dotnet/api/system.threading.tasks.taskscheduler?view=net-9.0