

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

**YAZILIM YAŞAM DÖNGÜSÜNDE TESTİN ÖNEMİ VE BİR
TEST OTOMASYONUNUN GERÇEKLEŞTİRİLMESİ**

ESRA NALBANT

YÜKSEK LİSANS TEZİ

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Haberleşme Programı

Danışman

Doç. Dr. Hacı İLHAN

Şubat, 2020

T.C.
YILDIZ TEKNİK ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

**YAZILIM YAŞAM DÖNGÜSÜNDE TESTİN ÖNEMİ VE BİR
TEST OTOMASYONUNUN GERÇEKLEŞTİRİLMESİ**

Esra NALBANT tarafından hazırlanan tez çalışması 28.02.2020 tarihinde aşağıdaki jüri tarafından Yıldız Teknik Üniversitesi Fen Bilimleri Enstitüsü Elektronik ve Haberleşme Mühendisliği Anabilim Dalı, Haberleşme Programı **YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Doç. Dr. Hacı İLHAN
Yıldız Teknik Üniversitesi
Danışman

Jüri Üyeleri

Doç. Dr. Hacı İLHAN, Danışman
Yıldız Teknik Üniversitesi

Doç. Dr. Hamit TORPİ, Üye
Yıldız Teknik Üniversitesi

Dr. Öğr. Üyesi Erdoğan AYDIN, Üye
İstanbul Medeniyet Üniversitesi

Danışmanım Doç. Dr. Hacı İLHAN sorumluluğunda tarafımca hazırlanan Yazılım Yaşam Döngüsünde Testin Önemi ve Bir Test Otomasyonunun Gerçekleştirilmesi başlıklı çalışmada veri toplama ve veri kullanımında gerekli yasal izinleri aldığımı, diğer kaynaklardan aldığım bilgileri ana metin ve referanslarda eksiksiz gösterdiğimi, araştırma verilerine ve sonuçlarına ilişkin çarpıtma ve/veya sahtecilik yapmadığımı, çalışmam süresince bilimsel araştırma ve etik ilkelerine uygun davrandığımı beyan ederim. Beyanımın aksinin ispatı halinde her türlü yasal sonucu kabul ederim.

Esra NALBANT

İmza

İÇİNDEKİLER

KISALTMA LİSTESİ	v
ŞEKİL LİSTESİ	vi
TABLO LİSTESİ	vii
ÖZET	viii
ABSTRACT	x
1 GİRİŞ	1
1.1 Literatür Özeti	2
1.1.1 Yazılım Testine Giriş	2
1.1.2 Yazılım Testi Neden Önemlidir?	2
1.1.3 Yazılım Testinin Temelleri	2
1.1.4 Yazılım Test Metodolojileri	3
1.1.5 Yazılım Test Yaşam Döngüsü	7
1.1.6 Yazılım Test Seviyeleri	11
1.1.7 Yazılım Test Teknikleri	14
1.1.8 Yazılım Test Türleri	18
1.1.9 Selenium Test Otomasyon Aracı	19
1.2 Tezin Amacı	27
1.3 Hipotez	28
1.4 Tezin İçeriği	29
2 MATERYAL VE METOT	30
2.1 Materyal ve Metot	30
2.1.1 Uygulamaların Arayüzleri ve Modülleri	30
2.1.2 Test Senaryo Oluşturma Süreci	33
2.1.3 Manuel Yazılım Test Süreci	59
2.1.4 Yazılım Test Otomasyon Süreci	59
2.1.5 Selenium Web Driver Kullanılarak Otomatik Testlerin Optimizasyonu	76

3 SONUÇ VE ÖNERİLER	86
3.1 Sonuç ve Öneriler.....	86
3.1.1 Manuel Yazılım Test Sonuçları	86
3.1.2 Selenium Web Driver ile Uygulama Adına Göre Otomatize Edilen ve Edilemeyen Test Senaryoları.....	89
3.1.3 Selenium Web Driver Aracı Test Otomasyon Sonuçları	91
3.1.4 Test Otomasyonu ve Manuel Testin Karşılaştırılması	92
3.1.5 Selenium IDE ile Selenium Web Driver'ın Karşılaştırılması	94
3.1.6 Sonuçlar.....	95
KAYNAKÇA	96
TEZDEN ÜRETİLMİŞ YAYINLAR	97

KISALTMA LİSTESİ

DPD	Defect Profile Document
SDLC	Software Development Life Cycle



ŞEKİL LİSTESİ

Şekil 1.1	Waterfall (Şelale) Model	5
Şekil 1.2	V Model.....	5
Şekil 1.3	Agile (Çevik) Model	6
Şekil 1.4	Yazılım Test Seviyeleri	11
Şekil 1.5	Entegrasyon Testi	12
Şekil 1.6	Kullanıcı Testi	14
Şekil 1.7	Kara Kutu Test Yaklaşımı	16
Şekil 1.8	Beyaz Kutu Test Yaklaşımı.....	17
Şekil 1.9	Selenium IDE	22
Şekil 1.10	Unit Test Proje Oluşturma.....	25
Şekil 1.11	Solution Explorer	26
Şekil 1.12	Nuget Package Manager.....	26
Şekil 1.13	Selenium Web Driver	27
Şekil 2.1	Halkbank Arayüzü.....	31
Şekil 2.2	Vakıfbank Arayüzü	32
Şekil 2.3	TCMB Arayüzü.....	33
Şekil 3.1	Halkbank Kredili Mevduat Ekranı	90
Şekil 3.2	HTS022 Numaralı Otomasyon Test Senaryo Kodu	90
Şekil 3.3	Vakıfbank Kredi Hesaplama Ekranı.....	91
Şekil 3.4	VTS021 Numaralı Otomasyon Test Senaryo Kodu	91
Şekil 3.5	Test Maliyeti Koşum Sayısı Grafiği.....	93

TABLO LİSTESİ

Tablo 1.1 Halkbank Uygulamasına ait modül bazlı manuel test durumları.....	86
Tablo 1.2 Vakıfbank.com.tr modül bazlı manuel test durumları	87
Tablo 1.3 TCMB.gov.tr modül bazlı manuel test durumları.....	88
Tablo 1.4 Grup bazlı manuel test durumları	89
Tablo 1.5 Uygulama Adına Göre Otomatize Edilen ve Edilemeyen Senaryolar.....	89
Tablo 1.6 Selenium Web Driver Test Otomasyon Sonuçları.....	92



Yazılım Yaşam Döngüsünde Testin Önemi ve Bir Test Otomasyonunun Gerçekleştirilmesi

Esra NALBANT

Elektronik ve Haberleşme Mühendisliği Anabilim Dalı

Yüksek Lisans Tezi

Danışman: Doç. Dr. Hacı İLHAN

Yazılım yaşam döngüsünde test büyük bir öneme sahiptir. Yazılım testleri, bir yazılım projesinin kaynak kodu kadar değerlidir. Günümüzde, yazılım kalite beklentisinin artmasının yanı sıra, yetkin personel, zaman ve maliyet kısıtları nedeniyle yazılım testi giderek önem kazanmaktadır. Özellikle kullanıcı arayüzü, yazılımların son kullanıcıyla etkileşimde olan kısım olduğu için yazılım testlerinin daha detaylı ve hızlı yapılması beklenmektedir. Uzun vadede, sürdürülebilir yazılım testleri projenin maliyetini önemli ölçüde düşürür. Yazılım geliştirme maliyetinin çoğu, yazıldıktan sonra yazılımı değiştirmek için yapılan bakımdır. Bunun nedeni testlerin hızlı bir şekilde kırılan olması ve bakım yapmanın çok pahalı olmasıdır. Sistem uygulanmasındaki en ufak bir değişiklik, test senaryoların çoğunu değiştirmeyi gerektirir. Örneğin, bir butonu yeniden adlandırmak, test bloğunu bozar ve testleri düzeltmek çok zaman alır. Artan kalite ihtiyacına cevap verebilmek için yazılım otomasyon faaliyetlerine başlanmadan önce beklentilerin, hangi yaklaşım ve araçların kullanılacağını belirlemenin önemlidir. Bu tez çalışmasında herkesin erişim sağlayabildiği örnek web uygulamalar üzerinden test senaryoları çıkarılmıştır. Çıkarılan senaryolara göre manuel olarak fonksiyonel testler gerçekleştirilmiştir. Ayrıca Selenium yazılım aracı kullanılarak oluşturulan test

senaryolarına göre test otomasyon alıřması yapılmıřtır. Selenium yazılım araları detaylı olarak anlatılmıř, Selenium IDE ve Selenium Web Driver araları kullanılarak test otomasyon alıřması yapılmıř ve arasındaki farklar kıyaslanmıřtır. Senaryolar otomatize edilmiř ve her biri kendi ortamında alıřtırılmıřtır. Sonrasında ise tüm senaryolar eřzamanlı yürütülebilir hale getirilmıřtır. Ayrıca manuel ve otomasyon testi arasındaki farklar kıyaslanmıřtır.

Anahtar Kelimeler: Yazılım yařam döngüsü, yazılım testi, test otomasyonu, Selenium IDE, Selenium Driver.



The Importance of Testing in Software Development Life Cycle and The Implementation Of A Test Automation

Esra NALBANT

Department of Electronics and Communications Engineering

Master's Thesis

Advisor: Associate Professor Hacı İLHAN

Testing has a vital importance in software development life cycle. Software tests are as valuable as the source code of a software project. Nowadays, software testing is becoming more and more important due to the increase in software quality expectation as well as qualified personnel, time and cost constraints. Software testing is expected to be more detailed and faster, especially since the user interface is the part of the software that interacts with the end user. In the long run, sustainable software testing significantly reduces the cost of the project. Most of the cost of software development stems from the maintenance of the software after it is written. This is because the tests are fragile and the maintenance is too expensive. The slightest change in system implementation requires changing most of the test scripts. For example, renaming a button breaks the test block, and it takes a lot of time to correct the tests. In order to meet the increasing need for quality, it is important to determine which approaches and tools to use in order to start the software automation activities.

In thesis study, test scenarios have been developed over sample web applications that everyone can access. According to the scenarios, functional tests were performed manually. In addition, test automation was performed according to the test scenarios created by using Selenium software tool. Selenium software tools are explained in detail and test automation studies are performed by using Selenium IDE and Selenium Web Driver tools. And the differences between are compared. The interoperability of the automated scenarios in groups was ensured. In thesis process, time, human resource, advantage, disadvantage, limitations and usability differences of manual test and automation test processes were compared.

Keywords: Software lifecycle, software testing, test automation, Selenium IDE, Selenium Driver.



1

Giriş

Yazılım testi, Yazılım Geliştirme Yaşam döngüsünün (SDLC) en önemli aşamalarından biridir. Test işlemi, kaynak kodundaki, yazılımın işlevselliğinde hatalara yol açabilecek bugları bulmayı amaçlar. Test, yazılımın kalitesini artırır. Test dediğimiz taktirde aklımıza öncelikle manuel ve otomasyon testi gelmektedir. Manuel test, bir test uzmanının son kullanıcı rolünü oynadığı ve uygulamanın işlevselliğinin kullanıcı gereksinimlerine uygun olup olmadığını test ettiği bir yaklaşımdır. Test uzmanının test stratejisini planlaması gerekir. Test senaryoları manuel yazılmalıdır ve sonra test verileri hazırlanmalıdır. Bu test senaryoların yürütülmesi ve aynı zamanda elde edilen sonuçların beklenen sonuçlar ile eşleşen olup olmadığını kontrol de test uzmanı tarafından manuel yapılmalıdır. Bu nedenle, manuel test çok zahmetli ve zaman alıcı bir görevdir. Tüm görevler yalnızca test uzmanı tarafından yapılması gerektiği için hatalar ortaya çıkabilir. Test uzmanının uygulamanın işlevselliği hakkında bilgili ve test senaryolarını hazırlamak için çok yetenekli olması gerekir. Web testinde otomatik teknikler ise günümüzün rekabetçi ortamında ki şirketlere büyük faydalar sağlamaktadır. Test otomasyonu önceden tahmin edilmiş sonuçlarla gerçek sonuçların karşılaştırılması ve testlerin koşulmasını kontrol etmek için (test edilmiş yazılımdan farklı olan) belirli yazılımın kullanılmasıdır. Otomasyon testinde de test senaryoları öncelikle manuel olarak yazılır, ancak bu senaryoların yürütülmesi kullanılan araç aracılığıyla otomatikleştirilir. Otomasyon testi test senaryolarının birçok kez tekrar edilmesi gerektiğinde çok yararlıdır. Bu nedenle, organizasyon gereksinimleri sık sık değiştiğinde veya bazı senaryo setinin tekrar tekrar uygulanması gerektiğinde otomasyon testinin daha hızlı ve güvenilir olduğu görülür. Böylelikle insan kaynaklarına yapılan yatırım, test otomasyonu ile azalmaktadır.

Yazılım testleri, bir yazılım projesinin kaynak kodu kadar değerlidir. Uzun vadede, sürdürülebilir yazılım testleri projenin maliyetini önemli ölçüde düşürür. Ancak, özellikle çalıştırılabilir, sürdürülebilir yazılım testleri yazmak çok zordur. Yazılım geliştirme maliyetinin çoğu, yazıldıktan sonra yazılımı değiştirmek için yapılan bakımdır. Test otomasyonu da bir yazılım geliştirme projesidir. Bu nedenle her test

otomasyon projesinin bakım süreci de olmalıdır. Otomatik test senaryolarının bakım süresi manuel olarak testleri koştuktan bazen daha fazla bile olabilir.

1.1 Literatür Özeti

1.1.1 Yazılım Testine Giriş

Test, ürünün herhangi bir hatası olup olmadığını belirlemek için belirli bir ürünün değerlendirilmesi sürecidir. Yazılım Testi, bir sistemi manuel veya otomatik yollarla değerlendirme sürecidir. Belirtilen gereksinimleri karşıladığını veya beklenen ve gerçek sonuçlar arasındaki farklılıkları tanımladığını gösteren doğrulama işlemidir. Yazılım Testi, geliştirilen bilgisayar yazılımlarının doğruluğunu, eksiksizliğini, güvenliğini ve kalitesini belirlemeye yardımcı olmak için kullanılan işlemdir. Test, paydaşlar adına gerçekleştirilen ve ürünle ilgili kaliteye ilişkin bilgileri ortaya koymayı amaçlayan bir teknik araştırma sürecidir. Bu, hataları bulma amacıyla bir programı veya uygulamayı yürütme işlemini içerir, ancak bunlarla sınırlı değildir. Kalite kavramı soyut değildir; bazı kişiler için değerlidir. Bunu göz önünde bulundurduğumuzda, test tamamen bilgisayar yazılımının doğruluğunu kanıtlamak değildir; test, bir özelliğe karşı ürünün durumunu ve davranışını karşılaştıran bir eleştiri veya karşılaştırmayı içerir.

1.1.2 Yazılım Testi Neden Önemlidir?

Tamamlanabilen yazılım paketinin gereksinimler/özellikler tarafından tanımlanan beklentilere göre çalıştığını doğrulamak için yazılım testi yapılır. Genel amaç, var olan her yazılım hatasını bulmak değil, müşteriyi, kullanılabilirliği ve/veya bakımı olumsuz yönde etkileyebilecek durumları ortaya çıkarmaktır. Testin sağladığı avantajlar aşağıda belirtilmiştir;

- i. Hataları ortaya çıkarmak,
- ii. Kullanıcının sorunları algılamasını önlemek,
- iii. Yazılım hatası olmadığını kanıtlamak,
- iv. Yazılımın güvenilirliği hakkında bilgi edinmek,
- v. Müşteriler tarafından dava edilmemek için kullanılır.

1.1.3 Yazılım Testinin Temelleri

Test sırasında yazılım mühendisliği, yazılımı parçalara dönüştürmek için kullanılan bir dizi test senaryoları üretir. Test, geliştiriciler tarafından yapıcı yerine yıkıcı olarak görülebilecek yazılım sürecindeki bir adımdır. Yazılım mühendisleri genellikle yapıcı insanlardır. Test, önyargılı doğruluk kavramlarının üstesinden gelmelerini ve hatalar tespit edildiğinde çakışmalarla başa çıkmalarını gerektirir.

a. Test Hedefleri

Test hedefleri olarak hareket eden bir dizi kural şunlardır:

- i. Test, hataları bulma amacıyla kullanılan bir sorun yürütme işlemidir.
- ii. İyi bir test senaryosu ile keşfedilmemiş bir hata bulma ihtimali yüksek olacaktır.
- iii. Başarılı bir test senaryosu yeni bir hata ortaya çıkarır.

b. Test Bilgi Akışı

Testler işletilir ve test sonuçları olarak kabul edilen tüm sonuçlar beklenen sonuçlarla karşılaştırılır. Hatalı veriler tanımlandığında, hata gösterilir ve hatayı ayıklamaya başlar. Hata ayıklama yordamı, test yordamının en öngörülemez ögesidir. Beklenen ve gerçek sonuçları arasında yüzde 0,01' lik bir tutarsızlık olduğunu gösteren bir hatanın tanımlanması ve düzeltilmesi saatler, günler veya aylar sürebilir. Hata ayıklama sürecindeki belirsizlik, testin zaman güvenilirliğinin zor olmasına neden olur.

c. Test Senaryo Tasarımı

Yazılım testlerinin tasarımı zorlu bir süreç olabilir. Ancak yazılım mühendisleri genellikle sonradan akla gelen düşünce olarak testi görür. Testin amacı, en az zamanlama ve çabayla en çok hatayı bulma olasılığın en yüksek olmasıdır. Geliştiriciye sistematik bir test yaklaşımı sunan çok sayıda test senaryo tasarımı yöntemi geliştirilmiştir. Yöntemler, testlerin eksiksiz olmasını sağlayacak ve yazılımdaki hataları ortaya çıkarmak için en yüksek olasılığı sunan bir yaklaşım sunar.

1.1.4 Yazılım Test Metodolojileri

Yaygın olarak kullanılan test metodolojilerinden bazıları şunlardır:

- i. Waterfall model
- ii. V model

- iii. Spiral model
- iv. RUP model
- v. Agile model
- vi. RAD model

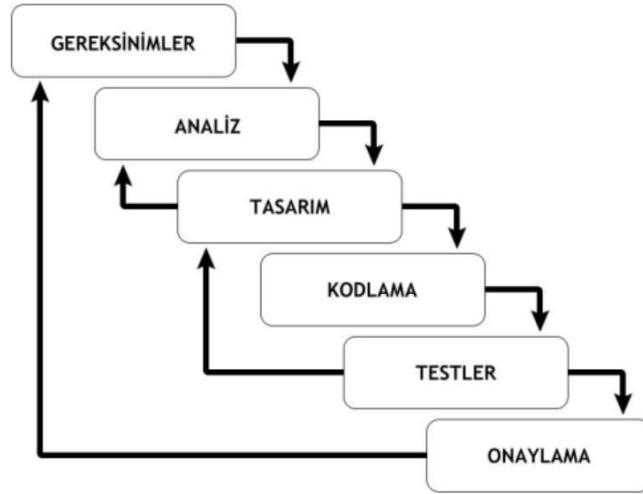
Bu metodolojilerin her birine tek tek bakalım [1].

a. Waterfall Model

Waterfall Modeli, yazılım geliştirme veya test için kullanılıp kullanılmadığına bakılmaksızın 'yukarıdan aşağıya' yaklaşımını benimser. Bu yazılım test metodolojisinde yer alan temel adımlar şunlardır:

- i. Gereksinim analizi
- ii. Test senaryo tasarımı
- iii. Test senaryo uygulaması
- iv. Test etme, hata ayıklama ve kodu veya ürünü doğrulama
- v. Dağıtım ve bakım

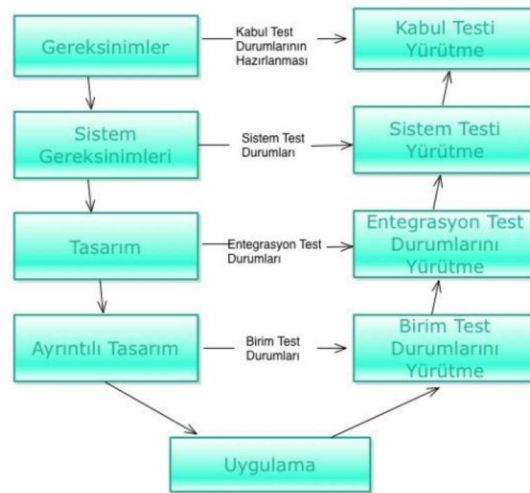
Bu metodolojide, yalnızca mevcut adımı tamamladıktan sonra bir sonraki adıma geçersiniz. Geriye veya ileriye atlamak veya aynı anda iki adım gerçekleştirmek için bir kapsam yoktur. Ayrıca, model yinelemeli olmayan bir yaklaşım izler. Bu metodolojinin temel yararı basit ve sistematik yaklaşımıdır. Ancak, test aşamasına ulaşılan kadar ve test etmeye kadar koddaki hatalar ve buglar bulunamadığı için birçok kusuru vardır. Bu genellikle zaman, para ve diğer değerli kaynakların israfına yol açabilir. Waterfall model şeması Şekil 1.1' de gösterilmiştir [1].



Şekil 1.1 Waterfall (Şelale) Model

b. V Model

V modeli, bu metodolojide yer alan farklı test süreci etkinliklerinin grafik gösteriminin 'V' harfine benzemesi gerçeğinden adını alır. Bu metodolojide yer alan temel adımlar Waterfall modeliyle aşağı yukarı aynıdır. Ancak, bu model hem 'yukarıdan aşağıya' hem de 'aşağıdan yukarıya' yaklaşımını izler. Bu metodolojiyi kullanmanın yararı, hem geliştirme hem de test faaliyetlerinin paralel yürütülebilmesidir. Örneğin, geliştirme ekibi gereksinim analizi etkinliklerine devam ettikçe, test ekibi aynı anda kabul test etkinliklerine başlar. Bu yaklaşımdan sonra, zaman gecikmeleri en aza indirilir ve kaynakların optimum kullanımı sağlanır. V modelinin şeması Şekil 1.2’ de gösterilmiştir [2].



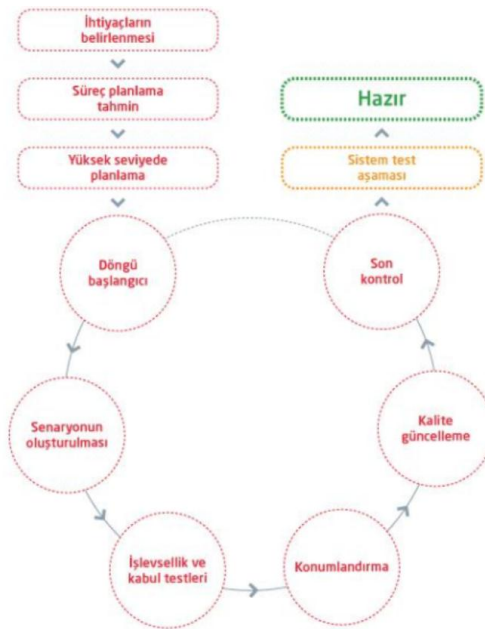
Şekil 1.2 V-model

c. Spiral Model

Sarmal modelde (spiral model) kısa yaşam zamanlarına sahip döngüler ile yazılım geliştirme tamamlanır. Her bir döngü sonunda yazılımın yeni bir uyarlaması hayata geçirilir. Amaç risklerin minimize edilmesidir. Her bir döngüde riskler (% olarak) ve bu riskler karşısında sistemin ne kadar etkileneceği irdelenir. Proje, önemli risklerin bertaraf edilmesiyle sonlanır. İlk döngü tamamlandıktan sonra, elde edilen ürün veya çıktının ayrıntılı bir analizi ve gözden geçirilmesi gerçekleştirilir. Belirtilen gereksinimlere veya beklenen standartlara uygun değilse, ikinci bir döngü izler ve böyle devam eder. Bu metodoloji yinelemeli bir yaklaşım izler ve genellikle karmaşık ve sürekli değişen gereksinimleri olan büyük projeler için uygundur.

d. Agile (Çevik) Model

Bu metodoloji ne tamamen sıralı bir yaklaşım ne de tamamen yinelemeli bir yaklaşım izler. Birçok ve yeni gelişimsel yöneme ek olarak her iki yaklaşımın seçici bir karışımıdır. Hızlı ve aşamalı gelişim bu metodolojinin temel ilkelerinden biridir. Odak noktası, sadece teorik süreçleri takip etmek yerine hızlı, pratik ve görünür çıktılar elde etmektir. Sürekli müşteri etkileşimi ve katılımı tüm geliştirme sürecinin ayrılmaz bir parçasıdır. Agile (çevik) model Şekil 1.3’ de gösterilmiştir [3].



Şekil 1.3 Agile (Çevik) Model

e. Hızlı Uygulama Geliştirme (RAD)

Metodoloji bileşen bazlı yapım prensibini kullanarak hızlı bir gelişimsel yaklaşım benimser. Verilen projenin farklı gereksinimleri kavradıktan sonra, hızlı bir prototip hazırlanır ve daha sonra beklenen çıkış koşulları ve standartları ile karşılaştırılır. Gerekli değişiklikler ve modifikasyonlar müşteri veya geliştirme ekibi ile (yazılım testi bağlamında) ortak görüşmeler sonrasında yapılır. Bu yaklaşımın kendi avantajları olsa da, projenin büyük, karmaşık ve doğasında son derece dinamik olması durumunda uygun olmayabilir, burada gereksinimleri sürekli değişir.

1.1.5 Yazılım Test Yaşam Döngüsü

Yazılım test yaşam döngüsü, yazılımın gerekli tasarım özelliklerini karşılayıp karşılamadığını doğrulamak için yapılması gereken çeşitli test faaliyetlerinden oluşur. Her kuruluştaki test, yazılım ürün geliştirme döngüsünde önemli aşamalardan biridir. Ancak, yürütülme şekli bir kuruluştan diğerine farklılık gösterir. Herhangi bir komplikasyonu önlemek için Yazılım Geliştirme Yaşam Döngüsü veya SDLC ile ilgili olarak, test işleminin ilk aşamalardan yürütülmesi tavsiye edilir [4]. Yazılım testi, SDLC'nin her aşamasını karşılayan kendi yaşam döngüsüne sahiptir. Yazılım test yaşam döngüsü diyagramı, çeşitli aşamalarını anlamalarına yardımcı olabilir. Bunlar:

- i. Gereksinim Aşaması
- ii. Test Planlama
- iii. Test Analizi
- iv. Test Tasarımı
- v. Test Doğrulama ve Yapım
- vi. Test Yürütme
- vii. Sonuç Analizi
- viii. Hata Takibi
- ix. Raporlama ve Yeniden Çalışma
- x. SonTest ve Uygulama
- xi. Uygulama Sonrası

Gereksinim Aşaması: Bu, yazılım yaşam döngüsü sürecini test eden ilk aşamasıdır. Bu aşamada geliştiriciler bir ürün tasarımı için gereksinimleri analiz etmede yer alır. Tester rolü de bu aşamada gereklidir, çünkü onlar geliştiricilerin düşünemediği 'kullanıcıların' bakış açısından düşünebilirler. Böylece, ürünün gereksinimlerini analiz etmek için

geliştiriciler, test uzmanı ve kullanıcılardan oluşan bir ekip oluşturulabilir. Yazılım gereksinim özelliklerini oluşturacak gereksinim dokümanı için ekip ile resmi toplantılar yapılabilir [4].

Test Planlama: Test planlama, ilerideki riskleri azaltmak için bir planı önceden belirlemek anlamına gelir. İyi tasarlanmış bir test plan dokümanı, sürece yönelik bir yaklaşım elde etmede önemli bir rol oynar. Projenin gereksinimleri onaylandıktan sonra, bir test planı dokümante edilir. Test planı yapısı aşağıdaki gibidir:

- i. Giriş: Bu test planının amacını açıklar.
- ii. Test Öğeleri: Bu dokümanı hazırlamak için gereken öğeler, proje Planı, Gereksinim dokümanı gibi dokümanlar burada listelenir.
- iii. Test edilecek özellikler: Bu, müşteriden gelen belirsiz ve açık gereksinimlere bağlı olarak test planının kapsama alanını, yani test edilecek özelliklerin listesini açıklar.
- iv. Test edilmemesi gereken özellikler: Test aşamasında atlanabilen kapsam dışı özellikler burada listelenmiştir. Eksik modüller veya düşük öneme sahip olanlar gibi test kapsamı dışında olan özellikler, örneğin, süreci engellemeyen GUI özellikleri listeye eklenebilir.
- v. Yaklaşım: Bu, planın düzeyine uygun olması gereken test stratejisidir. Bu planın daha yüksek ve daha düşük seviyeleri ile kabul edilmelidir.
- vi. Öge geçiş/başarısız ölçütleri: Kullanılan ölçütler, hangi test ögesinin geçtiğini veya başarısız olduğunu açıklamalı.
- vii. Askıya alma kriterleri ve devam etme koşulları: Askıya alma kriterleri, test faaliyetlerinin tamamını veya bir kısmını askıya almak için kullanılacak kriterleri belirtirken, yeniden başlama kriterleri testin askıya alınan kısmıyla devam edebileceği zamanı belirtir.
- viii. Testin teslim edilebilir olması: Bu, test süreci sırasında ve tamamlanmasından sonra düzenli olarak paydaşlara sunulması gereken dokümanların, raporların, grafiklerin bir listesini içerir.
- ix. Test yapma görevleri: Bu aşama, gerçekleştirilmesi gereken test görevlerini listeler. Bu, testlerin yürütülmesini, sonuçların değerlendirilmesi ve tasarlanan test planına göre onların dokümante edilmesini içerir. Bu,

kullanıcıların ve test uzmanların eksik fonksiyonlardan sakınması ve kaynak israfını önlemeleri için de yardımcı olur.

- x. Çevresel ihtiyaçlar: Uygulamanın tasarlanması gereken ortama bağlı olarak test planının özel gereksinimleri burada listelenmiştir.
- xi. Sorumluluklar: Bu aşama, risk durumunda sorumlu tutulabilecek kişilere sorumluluklar atar.
- xii. Personel ve eğitim ihtiyaçları: Uygulama/sistem ve kullanılacak test araçları ile ilgili eğitim, uygulamadan sorumlu personele açıklanması gerekir.
- xiii. Riskler ve olası durumlar: Bu, olası riskleri ve oluşabilecek çeşitli olayları ve bu gibi durumlarda neler yapılabileceğini vurgular.
- xiv. Onay: Bu, süreci kimin tamamlayacağına ve proje planının düzeyine bağlı olarak bir sonraki düzeye ilerlemesine izin vereceğine karar verir.

Test Analizi: Test planı dokümantasyonu yapıldıktan sonra, bir sonraki aşama SDLC'nin çeşitli aşamalarında ne tür yazılım testleri yapılması gerektiğini analiz etmektir.

Test Tasarımı: Test tasarımı, Gereksinim Dokümanı'nda belgelenen projenin gereksinimlerine göre yapılır. Bu aşama, manuel veya otomatik testin yapılıp yapılmayacağına karar verir. Otomasyon testinde, önce test için farklı yollar belirlenecek ve gerekirse test senaryolarının yazımı yapılacaktır. Test tasarım sürecinde projenin tüm özelliklerini kapsayan uçtan uca bir kontrol listesi gereklidir [4].

Test Doğrulama ve Yapım: Bu aşamada, test planı, test tasarımı ve otomatik test senaryoları tamamlanır. Stres ve performans testi planları da bu aşamada tamamlanır. Geliştirme ekibi birim kodlama yaptığında, test ekibinin bu birimi test etmede onlara yardımcı olması ve bulursa üründeki herhangi bir hatayı bildirmesi gerekir. Entegrasyon testi ve hata raporlaması, yazılım test yaşam döngüsünün bu aşamasında yapılır.

Test Yürütme: Bu aşamada çeşitli test senaryolarının planlanması ve yürütülmesi yapılır. Birim testi tamamlandıktan sonra, fonksiyonel testler bu aşamada yapılır. İlk başta, üst düzey hataları bulmak için üst düzey test yapılır ve hatalar anlık olarak çözülmesi için geliştirme ekibine hemen bildirilir. Test raporlarının düzgün bir şekilde belgelenmesi ve hataların geliştirme ekibine bildirilmesi gerekir.

Sonuç Analizi: Test senaryolarının başarılı bir şekilde yürütülmesinden sonra, test ekibinin beklenen değerleri gerçek değerlerle karşılaştırmak için yeniden test etmesi ve sonucu başarılı/başarısız olarak bildirmesi gerekir.

Hata İzleme: Bu, test uzmanı ve geliştiricinin hatayı yönetebilmesi için Hata Profil Dokümanı güncellenmesi gerektiğinden önemli aşamalardan biridir. Hata Profil Dokümanı aşağıdakileri içerir:

- i. Defect Id: Hatanın id'si.
- ii. Test Case Id: Bu hata için test örneği tanımlama.
- iii. Açıklama: Hatanın ayrıntılı açıklaması.
- iv. Özet: Bu alan, aranacak kayıt sayısını en aza indirmeye yardımcı olabilecek, hata hakkında bazı anahtar kelime bilgileri içerir.
- v. Defect Submitted By: Hatayı algılayan/bildiren tester'ın adı.
- vi. Gönderim Tarihi: Hatanın algılandığı ve bildirildiği tarih.
- vii. Yapı No: Gerekli test çalıştırma sayısı.
- viii. Sürüm No: Hatanın algılandığı ve düzeltildiği yazılım uygulamasının sürüm bilgileri.
- ix. Atanan: Hatayı düzeltmesi gereken developer'ın adı.
- x. Önem derecesi: Hatanın şiddet derecesi.
- xi. Öncelik: Hatayı düzeltme önceliği
- xii. Statü: Bu alan hatanın şimdiki statüsünü görüntüler.

Raporlama ve Yeniden Çalışma: Test yinelemeli bir işlemdir. Geliştirme ekibi tarafından bildirilen ve düzeltilen hata, bulunan hatanın çözüldüğünden emin olmak için test işleminden tekrar geçirilmesi gerekir. Yeniden test yapılmalıdır. Kalite Analisti ürünün hazır olduğundan emin olduktan sonra, yazılım üretim için piyasaya sürülür. Piyasaya sürülmeden önce, yazılım üst düzey testten bir tur daha geçmesi gerekir. Bu nedenle test devam eden bir süreçtir.

Son Test ve Uygulama: Bu aşama, kabul, yük, stres, performans ve kurtarma testleri gibi kalan test düzeylerine odaklanır. Uygulamanın Gereksinim Dokümanı ile ilgili

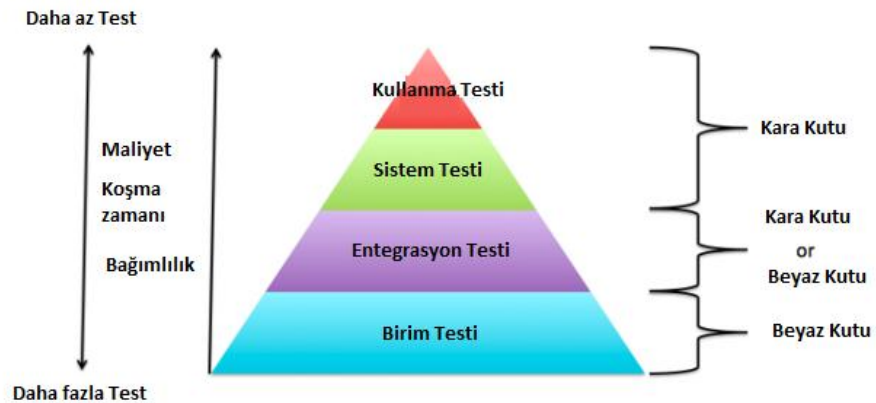
olarak belirtilen koşullar altında doğrulanması gerekir. Çeşitli belgeler güncellenir ve test için farklı matrisler, yazılım test yaşam döngüsünün bu aşamasında tamamlanır.

Uygulama Sonrası: Test sonuçları değerlendirildikten sonra, yazılım test yaşam döngüsünün çeşitli düzeylerinde meydana gelen hataların kaydı yapılır. Geliştirme ve iyileştirme planları oluşturmak devam eden bir işlemdir. Bu, gelecekteki projelerde benzer sorunların oluşmasını önlemeye yardımcı olur. Kısacası, gelecekteki uygulamalar için test sürecinin iyileştirilmesi için planlama bu aşamada yapılır [4].

1.1.6 Yazılım Test Seviyeleri

Test, bir sistemin veya sistemin bileşenlerinin belirtilen gereklilikleri karşılayıp karşılamadığını öğrenmek amacıyla yapılan bir değerlendirme sürecidir. Bu süreç, yazılım geliştirme uzmanları tarafından başlayan ve son kullanıcıya kadar uzanan teknik seviyelerden oluşur. Test seviyeleri temel olarak eksik alanları tanımlamak ve SDLC aşamaları arasındaki tekrarı önlemek içindir. Yazılım geliştirme yaşam döngüsü (SDLC); gereksinim analizi, tasarım, geliştirme, test ve bakım olmak üzere birçok aşamadan oluşmaktadır. Her aşama da testten geçer. Bu nedenle çeşitli test seviyeleri vardır.

Başlıca dört test seviyesi vardır. Bunlar; birim testi(unit testing), entegrasyon testi (integration testing), sistem testi(system testing) ve kabul testidir (acceptance testing) [5].



Şekil 1.4 Yazılım Test Seviyeleri

a. Birim Testi

Birim (unit) veya bileşen testi en temel test türüdür. Birim testi, yazılımın her bir bileşenini (modüller, programlar, sınıflar, nesneler vb.) izole ederek işleyişini kontrol eder. Ayrıca her bir bileşenin gereksinimleri yerine getirme ve işlevsellik açısından doğru olduğunu kanıtlamak için yapılır.

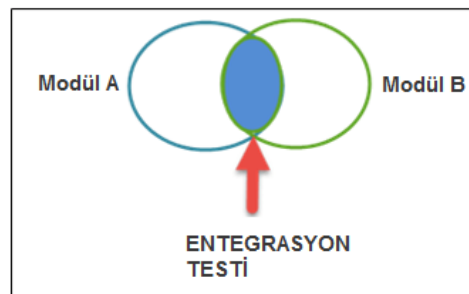
Birim test, genellikle yazılım geliştiriciler tarafından bir simülatör veya geliştirme ortamı yardımıyla yapılır. Ortaya çıkan hatalar kayıt altına alınmadan anında düzeltilir. Bu tür testler yazılım geliştirme sürecinin en erken aşamalarında yapılmalı ve yazılım, test ekibine bu şekilde teslim edilmelidir.

Bir yazılım geliştirme uzmanı, birim test yazmayı yazılım geliştirme sürecinin bir parçası olarak görmelidir. Birim test sayesinde hatalar, yazılım geliştirme süreci içerisinde tespit edilir. Böylece zaman ve paradan tasarruf sağlanır. SDLC döngüsündeki maliyetleri daha aşağı çeken BDD (Behaviour Driven Development) yaklaşımı da birim test yazmanın öneminin en büyük kanıtıdır. Kaliteli ürünler ortaya çıkarmak için birim test olmazsa olmazdır [5].

b. Entegrasyon Testi

Entegrasyon birleştirme demektir. Entegrasyon testi bir yazılımın bileşenlerinin birbirine entegre edilmesi sırasında yapılabileceği gibi iki farklı yazılımın birbirine entegre edilmesi sırasında da yapılabilir. Bu yüzden entegrasyon testi, birim entegrasyon testi ve sistem entegrasyon testi olarak farklı test seviyelerinde yapılabilir.

Yazılım geliştirme uzmanlarının birim test sırasında ayrı ayrı test ettikleri bileşenler birbirine entegre edildikleri zaman hataya sebep olabilirler. Entegrasyon testi, sistemin bu farklı bileşenlerinin(birimlerinin) birlikte doğru çalışıp çalışmadıklarını test etmeyi amaçlar [5].



Şekil 1.5 Entegrasyon Testi

c. Sistem Testi

Sistem testi, eksiksiz ve entegre bir sistemde gerçekleştirilir. Gereksinimlere göre sistemin uygunluğunun kontrol edilmesini sağlar. Bileşenlerin genel etkileşimini test eder. Yük, performans, güvenilirlik ve güvenlik testlerini içerir.

Sistem testi, sistemin şartnameye uygun olduğunu doğrulamak için yapılan son testtir. Bu yüzden test için hem fonksiyonel hem de fonksiyonel olmayan gereksinimler değerlendirir. Fonksiyonel gereksinimlerle ilgili sistem testi, test edilecek sistem için en uygun kara kutu (black box) teknikleri kullanılarak başlar. Ardından, beyaz kutu (white box) tekniklerine geçilerek kara kutu testlerinin yakalayamadığı hatalar yakalanabilir.

Sistemin çalışacağı ortamla ilgili hata riskini en aza indirmek için sistem testinde test ortamı mümkün olduğunca canlı ortama yakın olmalıdır [5].

d. Regresyon Testi

Regresyon testi uygulama ortamındaki yapılan tüm değişiklikleri; uygulamaya yeni eklenen özellikler, daha önceki yaşanan hataların düzeltilmesinden sonra, mevcut problemlerin giderildiği ve yeni yapılan güncellemelerin, eklenen özelliklerin yeni bir hata üretip üretmediğini kontrol amaçlı olarak yapılan yazılım test türüdür.

Uygulama ve uygulama ortamlarında gerekli değişiklikler ve sabitlemeler yapıldıktan sonra yeniden yapılan testlere çekilme (regresyon) testi denilir. Böylece, önceki testlerde belirlenen sorunların giderildiğinden ve yeni eklenen özelliklerin sorunsuz çalıştığından ve yeni hatalar oluşmadığından emin olunur.

Yazılım hataları ya da gerilemelerin (performans düşüklüğü), fonksiyonel ve fonksiyonel olmayan geliştirmelerin, sistemin bazı alanlarına yapılan yamaların, yapılandırma değişikliklerinin regresyon testi ile etkileri izlenebilir.

Regresyon Testleri çok önemlidir, çünkü günümüzde birçok yazılım yinlemeli geliştirim (İterative Development) süreci ile yapılmaktadır. Yinelemeli geliştirim sürecinde; her bir döngüde yeni bazı işlevler (Funcionality) eklenerek kısa döngüler kullanılmaktadır. Bu senaryoda; yeni eklenen işlevlerin var olan işlevleri bozmadığından emin olmak için, her bir döngüde Regresyon Testini uygulamak çok anlamlı olacaktır.

Sistem üzerinde yapılan her değişiklik istenmeyen yan etkiler doğurabilir. Her değişiklik ve hata düzeltilmesinin ardından regresyon testleri yapılarak, sistemin

bütünlüğü test edilir. Regresyon testlerinde sistem için kullanılan altyapı tanımlanmış bir duruma getirildikten sonra testler uygulanır.

Kurumsal dünyada yazılım geliştirmenin her aşamasında bu test uygulama türünü kesinlikle uyguluyor olmak gerekir. Çünkü günümüzde oluşturulan uygulamalar sürekli olarak gelişmektedir güncellenmektedir, müşterinin isteklerine göre değişiklikler yapılmaktadır. Aksi takdirde zaman ve maliyet açısından daha büyük kayıpların açığa çıkabilmesi kaçınılmaz olabilir.

Regresyon testi sadece geliştirmelerin amaca uygunluğunu değil aynı zamanda yazılım kalitesine de büyük katkılar sağlayabilir [6].

e. Kullanıcı Kabul Testi

Kabul testinin amacı, sisteme, sistemin parçalarına veya sistemin fonksiyonel olmayan gereksinimlerine karşı güven oluşturmaktır. Kabul testinde ana odak hataları bulmak değildir, sistemin canlıya hazır olduğunu göstermektir. Ancak ana odak hata bulmak olmasa bile yazım hatalarından uygulamada büyük bir soruna neden olabilecek hatalara kadar birçok hata açığa çıkabilir.

Kabul testleri genel olarak kullanıcı veya müşteri tarafından yapılır. Ancak, diğer hissedarlar da bu sürece dahil olabilir [5].



Şekil 1.6 Kullanıcı Testi

1.1.7 Yazılım Test Teknikleri

Yazılım test teknikleri, yazılım geliştirmede yazılan kodların kontrollerini yapmak ve fonksiyonların çalışma durumlarını sınamak için kullanılan test teknikleridir. Test teknikleri etkili bir şekilde test case tasarlamak için kullanılabilir.

Hata bulmak için kullanılan en önemli üç teknik; kara kutu testi, beyaz kutu testi ve gri kutu testi olarak belirtilmektedir [7].

a. Kara Kutu Testi

Test edilecek yazılımın iç işleyişi düşünülmeden yapılan testlere kara kutu testleri denilir. Bu yaklaşımda test edilecek yazılımın, sonucu bilinen bir davranışını doğrulamak için davranışın gerektirdiği girdi değerleriyle çalıştırılarak sınanır. Daha sonra yazılımın bu girdi karşısında elde edilen çıktısı, beklenen sonuçla karşılaştırılır. Bu yaklaşımın kara kutu olarak adlandırılmasının nedeni, testçinin testi uygularken yazılımın iç yapısıyla ilgilenmemesidir. Bu nedenle bu testler yazılım geliştiricilerden çok, test ekipleri tarafından uygulanır.

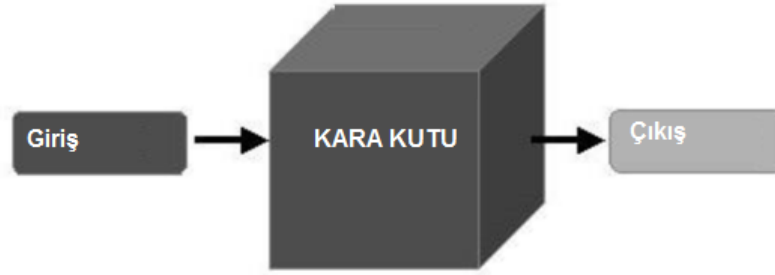
Kara kutu testinde bir yazılım parçası test ediliyorsa, test mühendisi bu yazılım parçasının girdisini ve buna karşılık sistem çıktısını bilir. Fakat bu çıktıya nasıl ulaşıldığıyla ilgilenmez. Çünkü kara kutu testlerinin amacı, verilen girdilerle istenilen çıktının elde edilmesidir. Bu nedenle yazılımın iç işleyişiyle ilgili diğer bilgilerle ilgilenilmez. Kara kutu test tekniğinde diğer önemli bir nokta da geliştirilen yazılımın tasarımından veya kodlanmasından kaynaklanabilecek hataların bulunmasıdır. Bu amaçla kara kutu testlerinin yapılması ve istenilen amaca ulaşılabilmesi için yazılım geliştiricilerle test ekibi birbirinden ayrı çalışmalıdır. Böylece test ekibi, tasarım ayrıntılarını bilerek yazılıma karşı ön yargı taşımaz.

Kara kutu test tekniğiyle geliştirilen yazılım içerisinde şu hata türleri tespit edilmeye çalışılır:

1. Doğru olmayan veya hiç olmayan işlevlerin tespiti
2. Arayüz hataları
3. Performans hataları
4. Veri tabanlarına ulaşma hataları veya veri yapılarındaki hatalar
5. İklendirme veya sonlandırma hataları

6. Sınır değ er hataları

Kara kutu test tekniđi basit olarak Őekil 'de g sterilmektedir.



Őekil 1.7 Kara kutu test yaklaşımı

Kara kutu testinin avantajları/dezavantajları

Kara kutu testinde yazılım mod llerinin i  iřleyiřiyle ilgilenilmez. Yazılımın belirlenen girdi karřısında beklenen  ıktıyı vermesi hedeflenir. Kara kutu testlerinin avantaj ve dezavantajları řunlardır:

Avantajlar:

- i. Yazılımlarda hataların bulunması i in etkin ve hızlı bir tekniktir.
- ii. Test durumları yazılırken gereksinimlerden hareket edildiđi i in gereksinimlerdeki tutarsızlıkların ve belirsizliklerin belirlenmesinde  nemlidir.
- iii. Test ilerin yazılımın ayrıntılarını bilmesine gerek yoktur.
- iv. Test iler ve kodcular birbirinden bađımsız  alıřabilirler.
- v. Test iler gereksinimleri dođrulamak ve gereken testleri ger ekleřtirmek i in yazılıma kullanıcı g z yle bakarlar. Bu da kodcular tarafından fark edilemeyen pek  ok olası hatanın ve eksiđin bulunmasına yardımcı olur.
- vi. Testleri yapacak kiřilerin sistem hakkında teknik ayrıntı bilmesine gerek yoktur.

Dezavantajlar:

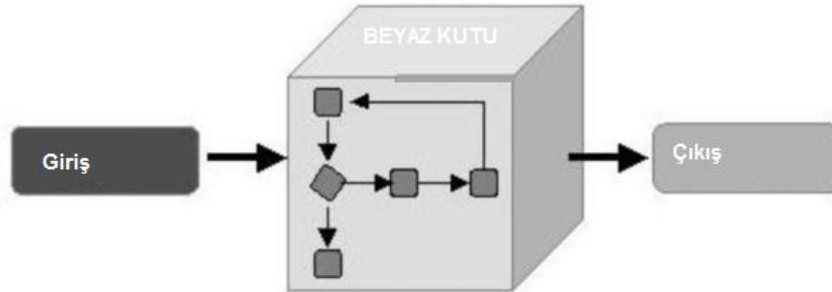
- i. Kara kutu testleri yazılımın belirli par asını hedeflemez. Bu nedenle bir ok hata tespit edilmeden kalabilir, bunlar i in bařka testler gerekir.
- ii. Sadece belirli sayıda girdi deđerleriyle testler yapılır. T m girdiler ile testlerin yapılması sonsuza kadar s rer.

- iii. Yazılım içerisinde bazı kod parçalarında birden fazla test yapılırken bazı kod parçaları hiç test edilmeden kalabilir.
- iv. Açık ve yalın olmayan gereksinimlerin test durumlarını tasarlamak ve testlerini yapmak kara kutu test tekniğinde zordur [8].

b. Beyaz Kutu Testi

Beyaz kutu testleri yazılımın iç yapısı bilinerek tasarlanır. Bu nedenle beyaz kutu testlerini gerçekleştirenler genellikle sistemin iç yapısını bilen kodculardır. Beyaz kutu testiyle programın iç yapısındaki birimlerin içindeki hatalar araştırılır. Kaynak kod, beyaz kutu testlerinin en önemli girdisi olduğundan, koda ulaşım olmadan beyaz kutu testleri yapılamaz. Bunun yanında risk analizleri ve tasarım kısıtları da beyaz kutu testlerinin planlanmasında, test stratejisinin belirlenmesinde, test araçlarının seçilmesinde ve test verilerinin oluşturulmasında kullanılır.

Beyaz kutu testleri veri, kontrol ve bilgi akışlarının, kodlama standartlarının, hata yakalama ve ayıklama yapısının analizlerini içerir. Beyaz (saydam) kutu test tekniği basitçe Şekil’de gösterilmektedir.



Şekil 1.8 Beyaz kutu test yaklaşımı

Beyaz kutu testinin avantajları/dezavantajları

Beyaz kutu test tekniğiyle sınanan birimin veya modülün belirlenen girdiyle beklenen çıktıyı vermesi hedeflenir. Ancak bu çıktıyı nasıl verdiği ve kod içinde hangi yollardan geçildiği de incelenir.

Avantajlar:

- i. Kod içerisinde gizli kalmış mantıksal hatalar bulunur.

- ii. Beyaz kutu testleriyle yazılan kodun optimizasyonuna katkıda bulunulur.
- iii. Kod içindeki fazla satırlar ayıklanarak ölü kod parçaları bulunur.
- iv. Kaynak kodun analiz edilmesi ve bu analize göre testlerin gerçekleştirilmesiyle yazılım içerisindeki hatalar daha erken aşamada ve daha hızlı bulunur.
- v. Yazılımın geliştirilmesi için belirlenmiş olan kodlama rehberine uyumluluk, tasarım kararlarına kodlama içerisinde uyulup uyulmadığı saydam kutu testleriyle net olarak görülür.
- vi. Beyaz kutu testleriyle yazılımcıların kod geliştirme yetenekleri desteklenir ve güçlendirilir.

Dezavantajlar:

- i. Eğer birim tümlleştirme testleri test ekibi tarafından yapılacaksa, bu iş için kodun iç yapısının bilinmesi gerekir. Bu da maliyeti arttırır.
- ii. Saydam kutu testleriyle sadece modül ve birimlerin iç işleyişleri test edildiğinden, tümlleştirmeden sonra ortaya çıkabilecek olan hatalar tespit edilemez [8].

c. Gri Kutu Testleri

Gri kutu testleri, saydam kutu ve kara kutu test tekniklerinin birlikte kullanılmasıdır. Bu testlerde gereksinimleri doğrulayacak test durumları kodun iç yapısı esas alınarak yazılır. Böylece gereksinimlerin gerçekleştiği doğrulandığı gibi, yazılımın iç yapısı sınanmış olur. Gri kutu testlerinde test ekibi yazılımın iç yapısını bildiğinden, yazılımın tasarımına ve kod yapısına karşı şartlanma olabileceğinden bazı hataları ortaya çıkartabilecek testler yapılmadan kalabilir [8].

1.1.8 Yazılım Test Türleri

Çeşitli yazılım test metodolojileri, ardışık yazılım test türleri boyunca rehberlik etmektedir. Test edilen uygulamanın gerçek işlevselliğini belirlemek için test çalışmaları geliştiricilere yardımcı olmak üzere tasarlanmıştır. Test çalışmaları, yazılımı test etme sürecinden geçilmesi için yönergeler sağlamaktadır. Yazılım testi, manuel ve otomasyon testi olmak üzere iki yazılım test türünü içerir.

a. Manuel Testi

Manuel test, bir test uzmanının son kullanıcı rolünü oynadığı ve uygulamanın işlevselliğinin kullanıcı gereksinimlerine uygun olup olmadığını test ettiği bir yaklaşımdır. Test uzmanının test stratejisini planlaması gerekir. Test senaryoları manuel yazılmalıdır ve sonra test verileri hazırlanmalıdır. Bu test senaryolarının yürütülmesi ve aynı zamanda elde edilen sonuçların beklenen sonuçlar ile eşleşen olup olmadığını kontrol de test uzmanı tarafından manuel yapılmalıdır.

b. Otomatik Testi

Otomatik test, önceden tahmin edilmiş sonuçlarla gerçek sonuçların karşılaştırılması ve testlerin koşulmasını kontrol etmek için (test edilmiş yazılımdan farklı olan) belirli yazılımın kullanılmasıdır. Otomasyon testinde de test senaryoları öncelikle manuel olarak yazılır, ancak bu senaryoların yürütülmesi kullanılan araç aracılığıyla otomatikleştirilir.

1.1.9 Selenium Test Otomasyon Aracı

Yazılım testi, Yazılım Geliştirme Yaşam döngüsünün (SDLC) en önemli aşamalarından biridir. Test süreçlerini iyileştirmek ve kolaylaştırmak adına bugün piyasada birçok yazılım ve araç bulunmaktadır. Bu araçlardan biri de Selenium’ dur. Selenium farklı tarayıcılar ve platformlar arasında web uygulamaları için ücretsiz olarak (açık kaynak) kullanılan otomatik test paketidir.

Selenium, web sitelerinin bazı test adımlarını web tarayıcınız üzerinden otomatik şekilde oluşturmanızı ve işletmenizi sağlayan bir tool’dur. Örnek vermek gerekirse, web uygulamanızın login ekranının başarılı ve başarısız senaryolarının testi için Selenium kullanılarak login ekranının açılması, ekrandaki input alanlarının girilmesi, butona basılması gibi step’ lerin otomatizasyonu yapılmaktadır. Bu yöntem ile başka senaryolarıda otomatize ederek manuel test yerine Selenium ile otomasyon testi yapılabilir [9].

Selenium, test otomasyon dünyasının önemli ürünlerinden biri haline gelen, açık kaynaklı, yazılım test otomasyon kütüphanesidir. Selenium tek bir araç değil, paketlerin oluşturduğu bir kütüphanedir. Bu sebeple Selenium Suite olarak adlandırılır.

Selenium, birçok işletim sistemini, tarayıcıyı ve yazılım dilini desteklemektedir. Netflix, Google, HubSpot gibi büyük şirketlerde aktif olarak kullanılmaktadır. Suite'e ait paketler, farklı test problemleri ve farklı ihtiyaçlar için çeşitli çözümler sunmaktadır.

Desteklenen İşletim Sistemler: Windows, MacOS, Linux, Unix

Desteklenen Tarayıcılar: IE, Firefox, Chrome, Opera, Safari

Desteklenen Programlama Dilleri: Java, Python, C#, PHP, Ruby, Perl, JavaScript

a. Selenium Avantajları

Açık Kaynak Olması: Selenium'un en büyük avantajı, ücretsiz ve taşınabilir (portable) bir kütüphane olmasıdır. Doğrudan bir maliyete sahip değildir. Kütüphane ücretsiz olarak indirilebilir ve kullanılabilir. Hali hazırda yazılmış olan kodlar, herhangi bir ücret ödemedi kullanılabılır ve değıştirilebilir.

Çoklu OS Desteęi: Selenium, Windows, MacOS, Linux, UNIX, vb. birçok İşletim Sisteminde (OS) çalışabilir ve bu sistemlerde geliştirilen senaryoları destekleyebilir. Örneęin, Windows işletim sistemi kullanarak test senaryoları oluşturabilir ve bu senaryoyu Linux tabanlı bir sistemde kolaylıkla çalıştırabiliriz.

Çoklu Tarayıcı Desteęi: Selenium; Internet Explorer, Chrome, Firefox, Opera, Safari, vb. birçok tarayıcıda destek sağlar. Bu sayede, testleri yürütürken aynı kod parçası ile birçok tarayıcıyı test etme imkanı sunar.

Paralel Test Yürütme: Bir web uygulamasını test ederken, uygulamanın farklı tarayıcılarda test edilmesi gerekebilir. Selenium WebDriver ve TestNG kullanarak bunu başarabiliriz. Yürütülecek fazla sayıda komut dosyası varsa ve bunları her bir tarayıcıda sırayla yürütmek gerekiyorsa, bu işlem hem zaman alıcı olur hem de tekrara düřtüęü için sıkıcı hale gelir. Bu sorun, Selenium'un Paralel Test Yürütme fonksiyonu ile önlenebilir.

Framework Desteęi: Selenium açık kaynak kodlu bir yazılım kütüphanesi olduęu için, piyasada bulunan Framework'ler tarafından rahatlıkla entegre edilebilir haldedir. Bu özellięi sayesinde yazılım geliştirme ve test dünyasında kullanılan önemli frameworkler (JUnit, TestNG, RobotFramework, NUnit vb) Selenium tarafından desteklenmekte ve gün geçtikçe yenileri, desteklenen frameworkler listesine eklenmektedir.

b. Selenium Dezavantajları

Yalnızca Web Testi Yapılabilmesi: Selenium, yalnızca web tabanlı uygulamaları desteklemektedir. Bu durum bazı kurumsal test otomasyon araçlarının karşısında Selenium'u geriye atmaktadır. Selenium kullanarak herhangi bir işletim sistemi ve tarayıcıda testler yapılabilir, ancak yalnızca Selenium kullanarak mobil testler yapılamamaktadır. Bunun için ek bir çözüm olarak WebDriver protokolünü kullanarak iOS ve Android mobil ve karma uygulamalar için Appium geliştirilmiştir.

API Test Desteği Olmaması: Test senaryolarının otomatize edilmesinde Kullanıcı Arayüzü (UI) testleri ne kadar önemliyse, API testleri de aynı derecede öneme sahiptir. Çeşitli otomasyon araçlarında bulunan bu özellik Selenium tarafından desteklenmemekte, yalnızca ek çözümler getirilerek API testleri yapılabilir.

Resmi Kontak Olmaması: Firmalar, test senaryolarını otomatize etmek için kullandığı uygulamalara belirli bir yatırım yapmaktadır. Bu yatırımın karşılığında, iletişime geçilecek resmi (official) bir kontakın olmayışı sorunlara yol açabilmektedir. Selenium açık kaynak bir paket olması sebebiyle gönüllü çalışma prensibine sahiptir.

Öğrenme Zorluğu: Günümüzün otomatikleştirilmiş test eğilimlerinden biri de kodsuz testtir (Record and Play). Bu yaklaşım, programlama bilgisi olmayan herkesin test otomasyonu yapabilmesine izin verir. Selenium, kod yazılmadan test yapılmasına izin vermemekte (Selenium IDE dışında) doğrudan kodlama üzerine odaklanmaktadır.

Raporlama Eksikliği: Otomatik olarak oluşturulan raporların olmayışı, Selenium'un en büyük eksikliklerinden biridir. Selenium'daki test hatalarını yakalamak için, hata anında ekran görüntüsü ya da log alınamaması ekibin sorunu hızla belirlemesi ve müdahale etmesini olumsuz yönde etkilemektedir. Farklı satıcılar/geliştiriciler ve hali hazırda oluşturulmuş frameworkler ile bu sorun aşılabilsede Selenium kişi (third party) çözümlere güvenmek zorundadır.

Resim Tabanlı Testlerde Yetersizlik: Selenium, diğer kurumsal otomasyon araçlarına kıyasla görüntü/resim doğrulama işleminde zorluk çekmektedir. Aslında bu durum birçok otomasyon aracında probleme sebep olmaktadır. Resim tabanlı testlerin yapılabilmesi için, geliştiriciler tarafından bu senaryolar için öngörülmuş tasarım/geliştirme çalışmaları yapılması gerekmektedir.

Kurulum Zorluğu: Test otomasyonu için kullanılan bazı araçlar kurulum konusunda fazlasıyla kullanıcı dostudur. Benzer amaçla kullanılan otomasyon uygulamaları,

uygulama kurulum dosyası yardımıyla rahatlıkla kurulup kullanılabilir. Selenium'un kurulumu ise daha fazla işlem gerektirmektedir.

c. Selenium Bileşenleri

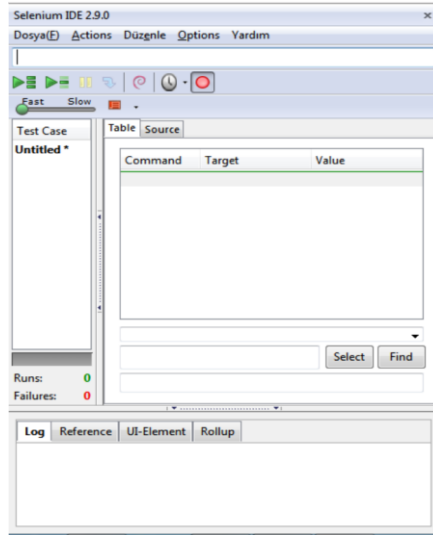
Selenium bileşenleri 4 ana başlıktan oluşur. Bunlar aşağıdaki gibidir:

- I. Selenium Integrated Development Environment (IDE)
- II. Selenium Remote Control (RC)
- III. Selenium WebDriver
- IV. Selenium-Grid

I. Selenium IDE

Selenium IDE, geliştirdiğimiz web tabanlı uygulamaların test edilmesini sağlayan ve bunları daha sonrasında otomatize ederek tarayıcı üzerinde aynı caseleri tekrar tekrar girmeden kontrol etmemizi sağlayan araçtır. Selenium paketi öğrenmesi en kolay olanıdır. Diğer eklentileri ile kolayca yükleyebileceğiniz Chrome ve Firefox eklentisidir. Ancak, sadeliği nedeniyle, Selenium IDE sadece bir prototip aracı olarak kullanılmaktadır.

Selenium IDE'nin exe'si çalıştırılmasının ardından Chrome veya Firefox'u browserları üzerinden Tools menüsünü kullanarak Selenium IDE yapısı kolayca öğrenilmektedir. Selenium IDE'yi açınca aşağıdaki ekran açılmaktadır. Selenium IDE üzerindeki menüler yardımıyla web uygulamanız için hazırladığınız test senaryolarını işletip kaydedebilir, akabinde işlettiğiniz senaryoları Selenium üzerinden tekrarlayarak testlerinizi devam edebilirsiniz. Selenium IDE tool'undaki alanlar kısaca aşağıda ifade edilmiştir [9].



Şekil 1.9 Selenium IDE

Menüler: Menü kısmında yeni test senaryosu oluşturmak, işletilen testleri record etmek ve herhangi bir senaryoyu farklı dilde export etme imkanı sağlamaktadır.

Table: Tüm işlemlerin saklandığı alandır.

Source: Table’da yer alan değerlerin HTML kodudur.

Play: Otomatize edilen caselerin çalıştırıldığı bölümdür.

Test Case: Gerçekleştirilen testlerin adlandırılarak tutulduğu alandır.

Command: Test senaryoları için uygun alanların girildiği bölümdür. Sık kullanılan command’lar aşağıda yer almaktadır.

open: URL’i açmak için kullanılır.

click: Bir objeye tıklamak için kullanılır.

clickAndWait: Objeye tıkladıktan sonra sayfanın yüklenmesini beklemek için kullanılır.

assertElementPresent: Belirlenen elementin yüklenmesini beklemek için kullanılır.

assertTextPresent: Belirlenen bir text’in ekranda olup olmadığını kontrol etmek için kullanılır.

deleteCookie: Belirlenen Cookie’nin delete edilmesi için kullanılır.

mouseOver: Mouse’u istenilen elemanın üzerine gitmesini sağlar.

verifyTextPresent: Belirlenen nesnenin belirmesini bekler ve sağlamasını yapmak için kullanılır.

waitForPageToLoad: Sayfanın yüklenmesini beklemek için kullanılır.

captureEntirePageScreenshot: Test yaptığınız sayfanın ekran görüntüsünü çerçeve olmadan kaydedilmesini sağlar.

waitForElementPresent: Sayfadaki bir elementin yüklenmesini beklemek için kullanılır.

type: Bir input'un içine yazı yazmak için kullanılır.

Target: Ekranda test edilecek alanın belirtildiği bölümdür.

Value: Yapılmak istenen test için değerin girildiği alandır.

Runs&Failures: Testlerin başarılı ve başarısız sayısının listelendiği bölümdür.

Record: Kayıt etmek için kullanılır.

Base URL: Testlerin hangi sayfada yapılacağını gösteren bölümdür.

Log: Yapılan testlerin loglarının yer aldığı bölümdür. Play kullanılarak gerçekleştirilen testler ile ilgili hata mesajlarının görüntülendiği bölümdür.

Reference: Command alanında kullanılan kodların dökümanate edildiği bölümdür.

II. Selenium Remote Control (Selenium RC)

Selenium RC, uzun bir süre için tüm Selenium projesinin en iyi test framework'üydü. Bu, kullanıcıların tercih ettikleri bir programlama dilini kullanmalarına izin veren ilk otomatik web test aracıdır. Şu anda, Selenium RC ve WebDriver, Selenium 2 oluşturmak için tek bir framework halinde birleştirilmiştir. Selenium 1, bu arada, Selenium RC'yi ifade ediyor [10].

Sürüm 2.25.0 itibariyle, RC aşağıdaki programlama dillerini destekleyebilir:

* Java

* C#

* PHP

* Python

*Perl

*Ruby

Özellikleri:

*Selenese'den daha etkileyici bir dil kullanarak bir test tasarlamak,

*Testinizi farklı işletim sistemlerinde farklı tarayıcılara (HtmlUnit hariç) karşı çalıştırmak,

*Uygulamanızı JavaScript'i destekleyen yeni bir tarayıcıyla karşı test etmek,

*Karmaşık AJAX tabanlı senaryolarla web uygulamalarını test etmek için kullanılır [10].

III. Selenium Web Driver

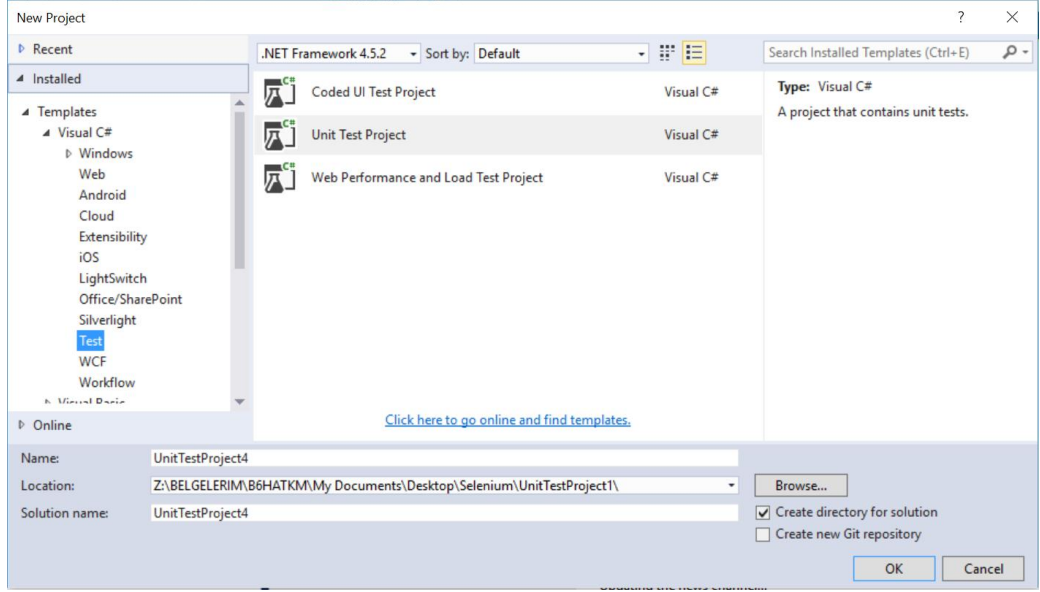
Selenium'un en önemli özelliklerinden biri Selenium RC olarak gösteriliyordu. WebDriver'ın eklenmesinin ardından Selenium 2.0 olarak yayınlanmıştır. Web Driver, Selenium'un kendi içinde geliştirmiş olduğu en önemli özelliklerinden biri olduğu düşünülmektedir. Web Driver, test yapan kişilerin isteklerine göre tarayıcının kendi metodlarını kullanarak işlem yapabildiğinden Web Driver birçok tarayıcıyı desteklemektedir (Firefox, Chrome, Opera, Safari, Internet Explorer). Bu önemli bir avantaj olup, test uzmanları tarafından tarayıcıların davranışlarını gözlemleyerek önlem alınmasını sağlar. Web Driver'ın geliştirilmesindeki en büyük hedef dinamik web sitelerinin içeriklerinin belli bir kısmının değişimi sırasında tüm sayfanın tekrardan yüklenmeden yapılan değişiklikleri görerek test senaryolarınızı devam ettirmenizde yardımcı olmaktır. Web Driver kullanmak için ihtiyaç duyulan Web Driver API'ı indirmektir. API ile istediğiniz dilde (Java, .NET, Python, Ruby, PHP, Perl gibi) test senaryolarınız oluşturabilirsiniz.

Visual Studio ve Selenium Web Driver

C# dilini kullanarak uygulama ve test geliştirebileceğimiz IDE, Visual Studio'dur. Microsoft güncel Visual Studio sürümünü kendi sitesinden ücretsiz olarak (Community versiyonu) indirilebilmektedir. Visual Studio üzerinden yazdığımız test otomasyonunu Web de görmemiz gerektiği için tercihe bağlı olarak Chrome veya Firefox'u bilgisayarınıza yüklemeniz gerekmektedir. Bu çalışmada Chrome üzerinden gideceğimiz için onu indirdik.

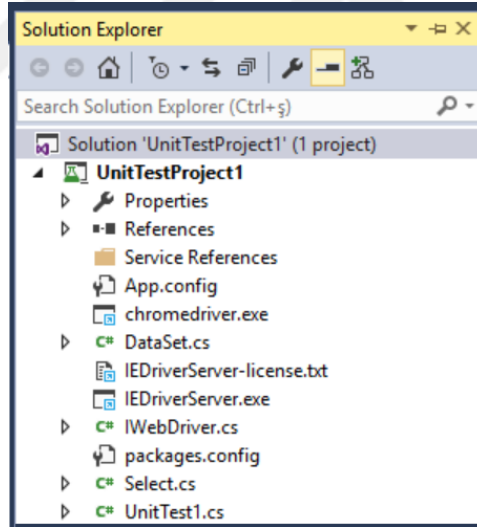
Visual Studio İle Test Projesi Oluşturmak

Visual Studio'yu açtıktan sonra, File>New>Project şeklinde adımları izliyoruz. Açılan ekranda bir sürü template bizi karşılıyor. Biz Unit Test Project'i seçip oluşturuyoruz. Oluştururken Name kısmına proje ismini, Location kısmında da projenin oluşturulacağı yeri belirliyoruz.



Şekil 1.10 Unit Test Projesi Oluşturma

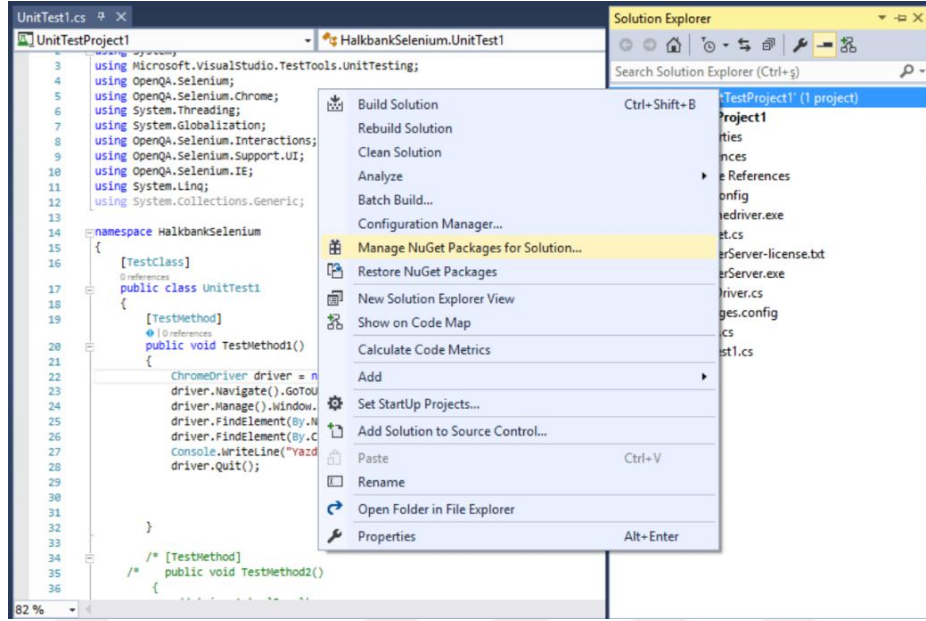
Oluşturulan projenin içerisinde nelerin olduğunu sağda bulunan Solution Explorer kısmına tıklamak gerekir. Eğer sağda Solution Explorer kısmı yoksa, Ctrl+Alt+L tuşlarına basıldığında gelecektir.



Şekil 1.11 Solution Explorer

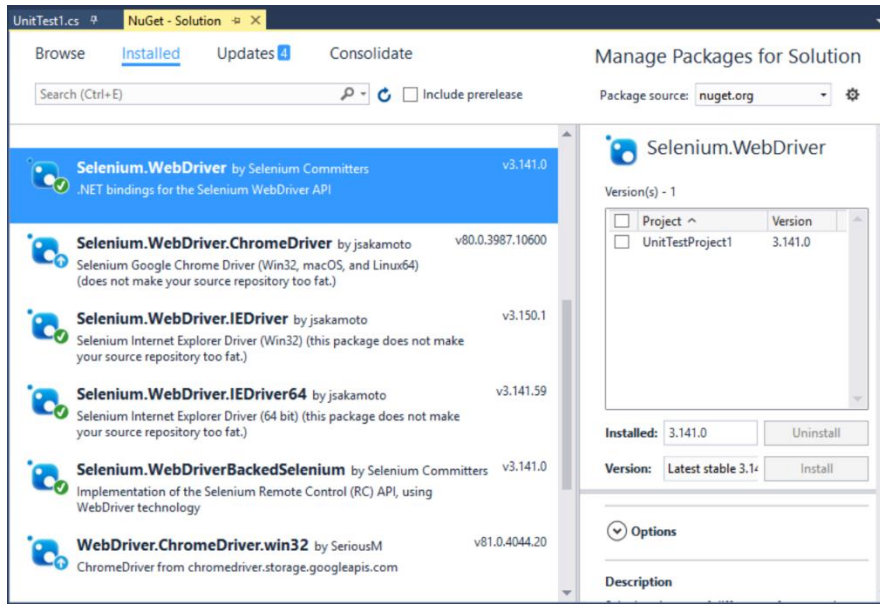
Selenium WebDriver'ı Projeye Ekleme

Oluşturduğumuz proje test projesi ancak içerisinde Selenium kurulu bir şekilde gelmemektedir. Visual Studio içerisinde Selenium'u eklememiz için Nuget Package Manager'i kullanmak gerekecektir.



Şekil 1.12 Nuget Package Manager

Tıkladığımızda Browse kısmından Selenium’u aratıp indirecektir.



Şekil 1.13 Selenium WebDriver

Bunları sırayla indirildiğinde artık proje de Selenium kurulu olacaktır. Bundan sonra artık test yazarken Selenium WebDriver’ı kullanabileceğiz ve gezinmeyi de Google Chrome üzerinden gerçekleştirebileceğiz. Selenium.Support ise yazacağımız kod için bazı konularda bize yardımcı olacak bir referans diyebiliriz.

IV. Selenium-Grid

Selenium’ un geliřtirdiđi ve geliřimi hala sren Selenium-Grid farklı sunucular zerinde farklı tarayıcılar ile paralel bir řekilde alıřtırılmaktadır. Selenium Grid ile farklı iřletim sistemi, donanım, cihaz gibi kombinasyonlar zerinde test sonularını grebilmek, dađıtık ortamda paralel olarak test srelerini yrtmek ve test sonularını hızlıca alabilmek amalanmıřtır. Bu testler paralel olarak iřletildiđinde sre olarak ciddi bir zaman kazancı oluřmaktadır. Selenium Grid, Selenium Hub ve Node yapısını kullanmaktadır.

1.2 Tezin Amacı

alıřmamızın temel amacı, web testlerinde zaman alan manuel grevleri azaltmaktır. Bu amaca ulařmak iin, ncelikle hangi test aracı kullanılacađı belirlendi. Bunun sonucunda test otomasyon dnyasında nemli bir konuma sahip olan Selenium Suite kullanılmasına karar verildi. Selenium’un en byk avantajı, cretsiz ve tařınabilir (portable) bir ktphane olmasıdır. Dođrudan bir maliyete sahip deđildir. Ktphane cretsiz olarak indirilebilir ve kullanılabilir. Hali hazırda yazılmıř olan kodlar, herhangi bir cret demeden kullanılabilir ve deđiřtirilebilir. İkinci konu olarak, iř kuralları ve sistem uygulamasındaki deđiřiklikler retim kodunu deđiřtirdiđinden ve bu senaryoların buna gre gncellenmesi gerektiđi iin, teste harcanan zamanın neredeyse yarısının otomatik test senaryoların bakımı iin olduđu analiz edildi. Test otomasyonunun artan maliyeti nedeniyle otomasyon sreci gzden geirilerek, tm test maliyetinin yzde 10’ unu gememek zere bakım maliyeti olan test senaryoların yazılması amalandı. Bunun iin bir uygulamadaki tm test senaryolarını otomatize etmek yerine sadece regresyon setinin otomasyon senaryolarını ele almak gerektiđi sonucuna varıldı. Otomasyonun, “Kayıt ve Oynatma” eylemlerinden daha fazla olduđunu kabul ederek, test otomasyonu yazılım mhendisliđi olarak deđerlendirilmelidir. Yani test otomasyon projelerinin bařında, bakım maliyetleri yazılım projeleri gibi dřnlmelidir.

1.3 Hipotez

Sistem regresyon testleri sırasında, testlerin daha sık kořturulabilmesi gerektiđinde otomatik testler manuel testten daha dođru sonular dođurur. Regresyon testinde, uygulamada bir deđiřiklik yapıldıđında, bir uygulamada aynı test caseleri tekrar tekrar yrtlır. Bu gibi durumlarda, senaryoların yrtlmesini manuel gerekleřtirmek ok

zaman alabilir ve burada hata yapma riski de oldukça yükselir. Otomasyon testi için ise bir otomasyon test aracı kullanmak gerekir. Bu araçların bazıları oldukça maliyetlidir. Ayrıca bu test araçlarını kullanmak için teknik kodlama bilgisine sahip olmak gerekir.

İş kurallarındaki ve sistem uygulamasındaki değişiklikler nedeniyle gereksinimler sürekli olarak değişmekte ve bu nedenle üretim kodu değişmektedir. Bu durum test senaryolarının çoğunu değiştirmeyi gerektirir. Örneğin, bir düğmeyi yeniden adlandırmak, test bloğunu bozar ve testleri düzeltmek çok zaman alır. Test aşamasının verimliliğini artırmak için, gereksinimler değiştikçe yazılan otomasyon test senaryolarını güncellemek gerekecektir. Yazılan test senaryoları gereksinimlerle veya uygulamayla eşitlenmemişse, kullanıcılar testleri yürütmeyi durdurur veya güvenilirmez sonuçlar elde edilir. Testleri yeniden eşitlemek için, testleri yeni gereksinimlere veya yeni uygulamaya uyum sağlamak için değiştirmemiz gerekir. Test otomasyon çalışmasının en yaygın problemi, testlerin kırılgan olması ve test senaryolarının senkronizasyonunu sağlamak için bakım maliyetinin beklenenden bazen daha fazla olmasıdır. Test ekipleri beklenmeyen bakım maliyetleri nedeniyle test otomasyonundan vazgeçmek zorunda bile kalabilir.

Test otomasyonu projeleri sırasında sık sık aşağıdaki sorunlarla karşılaşılır ve test otomasyonu projelerinin başarısızlığa yol açabilir:

- Yönetim, test otomasyonuna bir yazılım geliştirme olarak davranmaz. Herkesin test edebileceği ve otomasyonun sadece kayıt ve oynatmadan ibaret olduğu düşünülür.
- Kalite ekibi, test otomasyonu için yanlış test çalışma kümesini seçer. Aynı zamanda, yönetim tüm test çalışmalarının test otomasyonunun yüzde 100' ünü hedeflemektedir.
- Kalite Mühendisleri, tek bir göreve odaklanmak yerine manuel ve otomasyon testi arasında zaman geçirirler.
- Otomasyon test senaryolarının bakımının ve yönetimin zor olduğu göz ardı edilir.
- Otomatik test senaryolarının geliştirilmesi, bakımı ve yönetimi genellikle deneyimsiz test uzmanları tarafından karşılanması beklenir ve bu da manuel testten daha fazla zaman ve kaynağa ihtiyaç duyulmasına sebep olur.

1.4 Tezin İeriđi

Bu alıřmada herkesin eriřim sađlayabildiđi rnek web uygulamalar zerinden ıkarılan senaryolara gre manuel olarak fonksiyonel testler gerekleřtirilmiřtir. Ayrıca Selenium yazılım aracı kullanarak oluřturulan test senaryolarına gre test otomasyon alıřması yapılmıřtır. Selenium yazılım araları detaylı olarak anlatılmıř, Selenium IDE ve Selenium Web Driver araları kullanarak test otomasyon alıřması yapılmıřtır ve arasındaki farklar kıyaslanmıřtır. Senaryolar otomatize edilmiř ve her biri kendi ortamında alıřtırılmıřtır. Sonrasında ise tm senaryolar eřzamanlı yrtlebilir hale getirilmiřtir. Manuel ve otomasyon testi arasındaki farklar kıyaslanmıřtır.



2.1 Materyal ve Metot

Bu bölümde herkesin erişim sağlayabildiği örnek web uygulamalar üzerinden test senaryoları çıkarılmıştır. Çıkarılan senaryolara göre manuel olarak fonksiyonel testler gerçekleştirilmiştir. Selenium IDE ve Selenium Web Driver yazılım araçları kullanılarak oluşturulan test senaryolarına göre test otomasyon çalışması yapılmıştır. Sonrasında Selenim Web Driver aracı kullanılarak otomatik testlerin optimizasyonu gerçekleştirilmiştir.

2.1.1 Uygulamaların Arayüzleri ve Modülleri

Bu bölümde herkesin erişim sağlayabildiği **Hakbank.com.tr**, **Vakifbank.com.tr** ve **TCMB.gov.tr** uygulamaları üzerinden yazılacak test senaryolara ilişkin modüllere değinilmiştir.

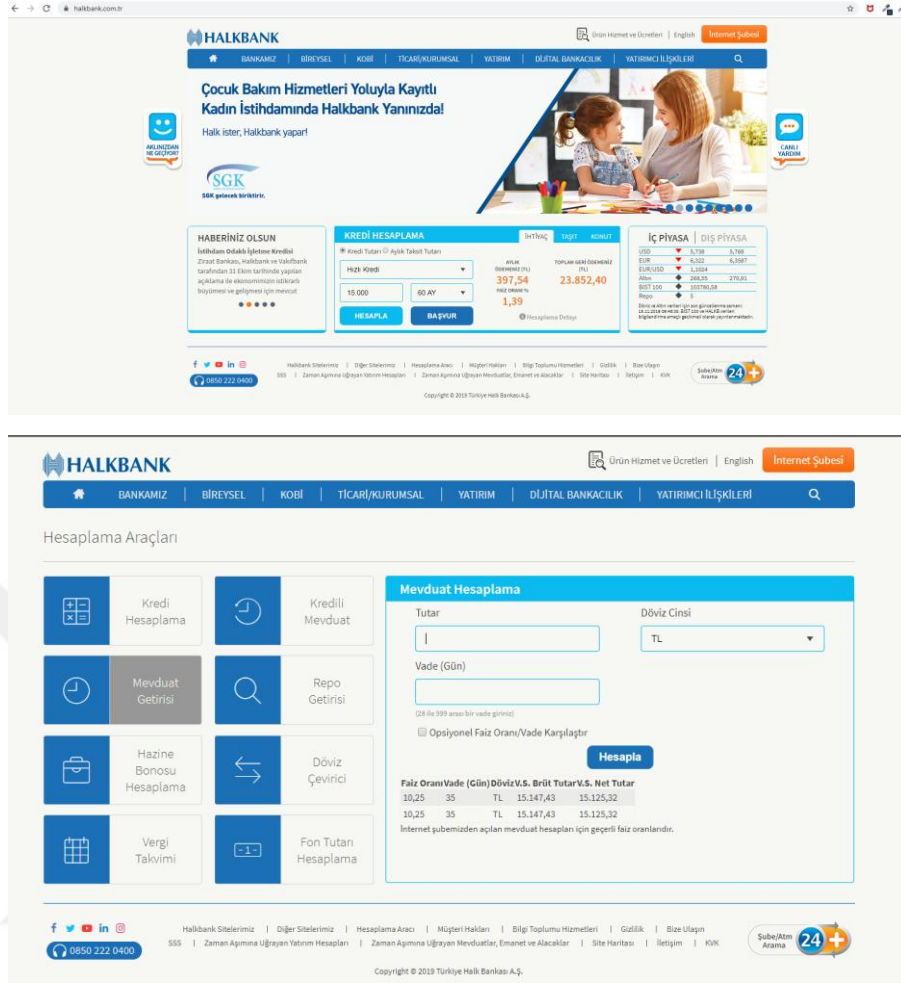
a. Halkbank Uygulama Arayüzü ve Modülleri

Bu başlık altında herkesin erişim sağlayabildiği **Hakbank.com.tr** uygulaması üzerinden yazılacak test senaryolara ilişkin modüllere değinilmiştir.

Yazılacak test senaryolarının daha okunabilir olması açısından her uygulama arayüzündeki modüller gruplanarak çalışılmıştır. Bu modüller aşağıda belirtilmiştir:

- İnternet Şubesi Modülü
- Kredi Hesaplama Modülü
- Hesaplama Aracı Modülü
- Kredi Mevzuat Modülü
- Mevzuat Getirisi Modülü
- Döviz Çevirici Modülü
- Fon Tutarı Modülü
- Hazine Bonosu Modülü
- Repo Getirisi Modülü

- Vergi Takvimi Modülü



Şekil 2.1 Halkbank Arayüzü

b. Vakıfbank Uygulama Arayüzü ve Modülleri

Bu başlık altında herkesin erişim sağlayabildiği **Vakıfbank.com.tr** uygulaması üzerinden yazılacak test senaryolara ilişkin modüllere değinilmiştir.

Yazılacak test senaryolarının daha okunabilir olması açısından her uygulama arayüzündeki modüller gruplanarak çalışmaya devam edilmiştir. Bu modüller aşağıda belirtilmiştir:

- İnternet Şubesi Modülü
- Hesaplama Aracı Modülü
- Mevduat Hesaplama Modülü
- Kredi Taksitleri Modülü
- Nakite Taksit Hesaplama Modülü

- Tahvil Bono Hesaplama Modülü

The image shows two screenshots of the VakıfBank website. The top screenshot is the homepage, featuring the VakıfBank logo, navigation tabs for 'Bireysel', 'KOBİ', 'Ticari', 'Kurumsal', 'Özel Bankacılık', and 'Yatırım', and a 'vibi' section with the tagline 'Dinleyen, anlayan, yapan zeka.' Below this are icons for 'Hesaplama Araçları', 'Kampanyalar', 'Bize Ulaşın', 'Sarı Rehber', and 'E-Finans'. The bottom screenshot shows the 'Mevduat Hesaplama Aracı' (Deposit Calculation Tool) interface. It includes a breadcrumb trail: 'Ana Sayfa > Hesaplama Araçları > Mevduat Hesaplama Aracı'. The main heading is 'Mevduat Hesaplama Aracı'. The form contains fields for 'Para Birimi' (TL), 'Tutar' (Amount), 'Vade Gün Sayısı' (Term in days), 'Vade Bitiş Tarihi' (Term End Date), and 'Faiz Oranı (boş bırakılabilir)' (Interest Rate (can be left empty)). A 'Hesapla' (Calculate) button is at the bottom right. A 'Linkler' (Links) section is at the bottom of the page.

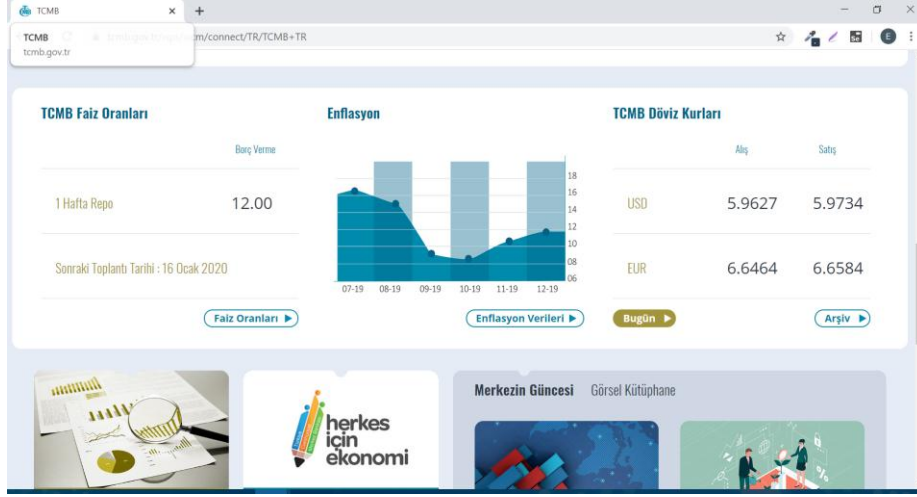
Şekil 2.2 Vakıfbank Arayüzü

c. TCMB Uygulama Arayüzü ve Modülleri

Bu başlık altında herkesin erişim sağlayabildiği **TCMB.gov.tr** uygulaması üzerinden yazılacak test senaryolara ilişkin modüle değinilmiştir.

Bu modül aşağıda belirtilmiştir:

- TCMB ile Halkbank Döviz Karşılaştırma Modülü



Şekil 2.3 TCMB Arayüzü

2.1.2 Test Senaryo Oluşturma Süreci

Bu bölümde sırasıyla herkesin erişim sağlayabildiği Halkbank.com.tr, Vakıfbank.com.tr ve TCMB. Gov.tr uygulamalara ait arayüzlerin ve modüllerin fonksiyonel özellikleri belirtilmiştir. Yapılan bu çalışmada belirlenen fonksiyonlara ait test senaryoları yazılmıştır. Test senaryo örnekleri aşağıda yer almaktadır.

a. Halkbank Uygulama Arayüzü Test Senaryoları

Halkbank uygulama arayüzünde bazı fonksiyonel özellikler belirlenip test senaryoları yazılmıştır. Halkbank uygulamasında oluşturulan test senaryoları 10 farklı başlık altında toplanmıştır. Halkbank uygulama arayüzüne ait test senaryoları HTS kodu ile başlamaktadır. Mesela HTS011 olarak gösterilen test senaryosu 11 numaralı test senaryosunu ifade etmektedir. Fonksiyonel özellikleri içeren test senaryoları her modülün altında ayrıntılı olarak belirtilmiştir.

Halkbank İnternet Şubesi Modülü

HTS001-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "İnternet Şubesi" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra butona tıklanarak doğru pencereye geçiş yapıp yapılmadığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi" butona tıklandığında doğru pencereye geçmesi.

HTS002-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi" butonuna tıklandıktan sonra çıkan pencerede "Bireysel" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi" butonuna tıklandıktan sonra çıkan pencerede "Bireysel" butona tıklandığında doğru pencereye geçiş yapması.

HTS003-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Müşteri/TC Kimlik Numarası" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Müşteri/TC Kimlik Numarası" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Müşteri/TC Kimlik Numarası" alanına tıklanması.

HTS004-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Parola" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Parola" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Parola" alanına tıklanması.

HTS005-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Giriş" butonuna tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonuna tıklanması.

HTS006-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi" butonuna tıklandıktan sonra çıkan pencerede "Kurumsal" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi" butonuna tıklandıktan sonra çıkan pencerede "Kurumsal" butona tıklandığında doğru pencereye geçiş yapması.

HTS007-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Firma Müşteri Numarası" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanır. "Kurumsal" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Firma Müşteri Numarası" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Kullanıcı Müşteri Numarası" alanına tıklanması.

HTS008-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Kullanıcı Müşteri Numarası" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanır. "Kurumsal" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Kullanıcı Müşteri Numarası" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Kullanıcı Müşteri Numarası" alanına tıklanması.

HTS009-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Parola" alanının test edilmesi.

Test Senaryo: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanır. "Kurumsal" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Parola" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Parola" alanına tıklanması.

HTS010-Test Senaryo: Halkbank.com.tr sayfasında "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanır. "Kurumsal" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Giriş" butonuna tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Şubesi", "Kurumsal" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonuna tıklanması.

Halkbank Hesaplama Aracı Modülü

HTS011-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "Hesaplama Aracı" linkinin test edilmesi.

Test Açıklaması: Ana menüdeki "Hesaplama Aracı" linkinin üstüne gelinir. Daha sonra linke tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "Hesaplama Aracı" linkine tıklandığında doğru pencereye geçmesi.

Halkbank Kredi Hesaplama Modülü

HTS012-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "Kredi Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Ana menüdeki "Kredi Hesaplama" yazısının üstüne gelinerek, Kredi Hesaplama alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Ana menüde "Kredi Hesaplama" alanının olması.

HTS013-Test Senaryo: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde "Kredi Tutarı" alanının test edilmesi.

Test Açıklaması: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde "Kredi Tutarı" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Hesaplama" menüsünde "Kredi Tutarı" alanına tıklanarak veri girilmesi.

HTS014-Test Senaryo: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde "Ay" alanının test edilmesi.

Test Açıklaması: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde "Ay" alanına tıklanarak combobox'tan veri seçimi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Hesaplama" menüsünde "Ay" alanına tıklanarak combobox'tan veri seçimi yapılması.

HTS015-Test Senaryo: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Kredi Hesaplama" menüsünde "Hesapla" butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Kredi Hesaplama" menüsünde "Hesapla" butonuna tıklanması.

HTS016-Test Senaryo: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde Kredi Hesapla işleminin yapılması

Test Açıklaması: Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde, "Kredi Tutarı" ve "Ay" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: " Kredi Hesaplama " menüsünde "Kredi Tutarı" ve "Ay" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

HTS017-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "İhtiyaç" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İhtiyaç" alanının üstüne gelinir. Daha sonra butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "İhtiyaç" alanına tıklandığında doğru pencereye geçmesi.

HTS018-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "Taşıt" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "Taşıt" alanının üstüne gelinir. Daha sonra butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "Taşıt" alanına tıklandığında doğru pencereye geçmesi.

HTS019-Test Senaryo: Halkbank.com.tr sayfasında ana menüdeki "Konut" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "Konut" alanının üstüne gelinir. Daha sonra butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "Konut" alanına tıklandığında doğru pencereye geçmesi.

Halkbank Kredili Mevduat Modülü

HTS020-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" butona tıklandığında doğru pencereye geçiş yapması.

HTS021-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Kredili Mevduat" butonuna tıklandıktan sonra çıkan pencerede "Kredi Tutarı" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi Tutarı" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredili Mevduat" menüsüne tıklandıktan sonra çıkan pencerede "Kredi Tutarı" alanına tıklanarak veri girilmesi.

HTS022-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Kredili Mevduat" butonuna tıklandıktan sonra çıkan pencerede "Kullanım Tarihi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Kullanım Tarihi" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredili Mevduat" menüsüne tıklandıktan sonra çıkan pencerede "Kullanım Tarihi" alanına tıklanarak veri girilmesi.

HTS023-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Kredili Mevduat" butonuna tıklandıktan sonra çıkan pencerede "Ödeme Tarihi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Ödeme Tarihi" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredili Mevduat" menüsüne tıklandıktan sonra çıkan pencerede "Ödeme Tarihi" alanına tıklanarak veri girilmesi.

HTS024-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Kredili Mevduat" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Hesapla" butonuna tıklanıp tıklanılmadığı kontrol edilir.

Beklenen Sonuç: "Kredili Mevduat" menüsü açıldıktan sonra "Hesapla" butonuna tıklanması.

HTS025-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Kredili Mevduat" butonuna tıklandıktan sonra çıkan pencerede Kredili Mevduat Hesapla işleminin yapılması

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi Tutarı", "Kullanım Tarihi" ve "Ödeme Tarihi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredili Mevduat" menüsüne tıklandıktan sonra "Kredi Tutarı", "Kullanım Tarihi" ve "Ödeme Tarihi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Halkbank Mevduat Getirisi Modülü

HTS026-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" yazısının üstüne gelinerek, Mevduat Getirisi alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" alanının olması.

HTS027-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Mevduat Getirisi" butonuna tıklandıktan sonra çıkan pencerede "Tutar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Getirisi" menüsüne tıklandıktan sonra çıkan pencerede "Tutar" alanına tıklanarak veri girilmesi.

HTS028-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Mevduat Getirisi" butonuna tıklandıktan sonra çıkan pencerede "Vade (Gün)" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade (Gün)" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Getirisi" menüsüne tıklandıktan sonra çıkan pencerede "Vade (Gün)" alanına tıklanarak veri girilmesi.

HTS029-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Mevduat Getirisi" butonuna tıklandıktan sonra çıkan pencerede "TL" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "TL" combobox'ından veri seçimi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Getirisi" menüsüne tıklandıktan sonra çıkan pencerede "TL" combobox'ından veri seçilebilmesi.

HTS030-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Mevduat Getirisi" butonuna tıklandıktan sonra çıkan pencerede “Hesapla” butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede “Hesapla” butonuna tıklanıp tıklanılmadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Getirisi" menüsü açıldıktan sonra “Hesapla” butonuna tıklanması.

HTS031-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Mevduat Getirisi" butonuna tıklandıktan sonra çıkan pencerede Mevduat Hesapla işleminin yapılması

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede “Tutar”, "Vade (Gün)" ve “TL” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Getirisi" menüsüne tıklandıktan sonra “Tutar”, "Vade (Gün)" ve “TL” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Halkbank Repo Getirisi Modülü

HTS032-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" butona tıklandığında doğru pencereye geçiş yapması.

HTS033-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Repo Getirisi" butonuna tıklandıktan sonra çıkan pencerede "Repo Vadesi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Repo Vadesi" alanına tıklanarak combobox'tan seçim yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Repo Getirisi" menüsüne tıklandıktan sonra çıkan pencerede "Repo Vadesi" alanına tıklanarak combobox'tan seçim yapılması.

HTS034-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Repo Getirisi" butonuna tıklandıktan sonra çıkan pencerede "Tutar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Repo Getirisi" menüsüne tıklandıktan sonra çıkan pencerede "Tutar" alanına tıklanarak veri girilmesi.

HTS035-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Repo Getirisi" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Hesapla" butonuna tıklanıp tıklanılmadığı kontrol edilir.

Beklenen Sonuç: "Repo Getirisi" menüsü açıldıktan sonra "Hesapla" butonuna tıklanması.

HTS036-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Repo Getirisi" butonuna tıklandıktan sonra çıkan pencerede Repo Getirisi Hesaplama işleminin yapılması.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Repo Vadesi" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Repo Getirisi" menüsüne tıklandıktan sonra "Repo Vadesi" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Halkbank Hazine Bonosu Hesaplama Modülü

HTS037-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsüne tıklanarak doğru pencereye geçiş yapıp yapılmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" butona tıklandığında doğru pencereye geçiş yapması.

HTS038-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Hazine Bonosu Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Bono Vadesi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Bono Vadesi" alanına tıklanarak combobox' tan seçim yapılıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Hazine Bonosu Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Bono Vadesi" alanına tıklanarak combobox' tan seçim yapılması.

HTS039-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Hazine Bonosu Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Ana Para" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Ana Para" alanına tıklanarak veri girişi yapılıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Hazine Bonosu Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Ana Para" alanına tıklanarak veri girilmesi.

HTS040-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Hazine Bonosu Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Hesapla” butonuna tıklanıp tıklanılmadığı kontrol edilir.

Beklenen Sonuç: "Hazine Bonosu Hesaplama" menüsü açıldıktan sonra “Hesapla” butonuna tıklanması.

HTS041-Test Senaryo: Halkbank.com.tr sayfasında “Hesaplama Aracı”, "Hazine Bonosu Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Hazine Bonosu Hesaplama işleminin yapılması.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonosu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Bono Vadesi” ve “Ana Para” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Hazine Bonosu Hesaplama" menüsüne tıklandıktan sonra “Bono Vadesi” ve “Ana Para” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Halkbank Döviz Çevirici Modülü

HTS042-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" butona tıklandığında doğru pencereye geçiş yapılması.

HTS043-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Döviz Çevirici" butonuna tıklandıktan sonra çıkan pencerede "Para Birimi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanır. Daha sonra çıkan pencerede "Para Birimi" alanına tıklanarak combobox’tan seçim yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Döviz Çevirici" menüsüne tıklandıktan sonra çıkan pencerede "Para Birimi" alanına tıklanarak combobox'tan seçim yapılması.

HTS044-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Döviz Çevirici" butonuna tıklandıktan sonra çıkan pencerede "Miktar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanır. Daha sonra çıkan pencerede "Miktar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Döviz Çevirici" menüsüne tıklandıktan sonra çıkan pencerede "Miktar" alanına tıklanarak veri girilmesi.

HTS045-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Döviz Çevirici" butonuna tıklandıktan sonra çıkan pencerede "Çevrileceği Para Birimi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanır. Daha sonra çıkan pencerede "Çevrileceği Para Birimi" alanına tıklanarak combobox'tan seçim yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Döviz Çevirici" menüsüne tıklandıktan sonra çıkan pencerede "Çevrileceği Para Birimi" alanına tıklanarak combobox'tan seçim yapılması.

HTS046-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Döviz Çevirici" butonuna tıklandıktan sonra çıkan pencerede "Çevir" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanır. Daha sonra çıkan pencerede "Çevir" butonuna tıklanıp tıklanılmadığı kontrol edilir.

Beklenen Sonuç: "Döviz Çevirici" menüsü açıldıktan sonra "Çevir" butonuna tıklanması.

HTS047-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Döviz Çevirici" butonuna tıklandıktan sonra çıkan pencerede Döviz çevirme işleminin yapılması.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Döviz Çevirici" menüsüne tıklanır. Daha sonra çıkan pencerede "Para Birimi",

"Çevrileceği Para Birimi" ve "Miktar" alanlarına veri girişi yapıp, "Çevir" butonuna tıklanarak hesaplama işleminin yapılıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Döviz Çevirici" menüsüne tıklandıktan sonra "Para Birimi", "Çevrileceği Para Birimi" ve "Miktar" alanlarına veri girişi yapıp, "Çevir" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Halkbank Vergi Takvimi Modülü

HTS048-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Vergi Takvimi" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Vergi Takvimi" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Vergi Takvimi" butona tıklandığında doğru pencereye geçiş yapması.

Halkbank Fon Tutarı Hesaplama Modülü

HTS049-Test Senaryo: Halkbank.com.tr sayfasında Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" butona tıklandığında doğru pencereye geçiş yapması.

HTS050-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Fon Tutarı Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Yatırım Fonu" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Yatırım Fonu" alanına tıklanarak combobox' tan seçim yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Fon Tutarı Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Yatırım Fonu" alanına tıklanarak combobox' tan seçim yapılması.

HTS051-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Fon Tutarı Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Pay Adedi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Pay Adedi" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Fon Tutarı Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Pay Adedi" alanına tıklanarak veri girilmesi.

HTS052-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Fon Tutarı Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Hesapla" butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Fon Tutarı Hesaplama" menüsü açıldıktan sonra "Hesapla" butonuna tıklanması.

HTS053-Test Senaryo: Halkbank.com.tr sayfasında "Hesaplama Aracı", "Fon Tutarı Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Fon Tutarı Hesapla işleminin yapılması

Test Açıklaması: "Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Yatırım Fonu" ve "Pay Adedi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Fon Tutarı Hesaplama" menüsüne tıklandıktan sonra "Yatırım Fonu" ve "Pay Adedi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

b. Vakıfbank İnternet Bankacılığı Modülü

Vakıfbank uygulama arayüzünde bazı fonksiyonel özellikler belirlenip test senaryoları yazılmıştır. Vakıfbank uygulamasında oluşturulan test senaryoları 6 farklı başlık altında toplanmıştır. Vakıfbank uygulama arayüzüne ait test senaryoları VTS kodu ile

başlamaktadır. Mesela VTS011 olarak gösterilen test senaryosu 11 numaralı test senaryosunu ifade etmektedir. Fonksiyonel özellikleri içeren test senaryoları her modülün altında ayrıntılı olarak belirtilmiştir.

Vakıfbank İnternet Bankacılığı Modülü

VTS001-Test Senaryo: Vakıfbank.com.tr sayfasında ana menüdeki "İnternet Bankacılığı" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: " İnternet Bankacılığı" butona tıklandığında doğru pencereye geçmesi.

VTS002-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı" butonuna tıklandıktan sonra çıkan pencerede "Bireysel" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki " İnternet Bankacılığı " butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı" butonuna tıklandıktan sonra çıkan pencerede "Bireysel" butona tıklandığında doğru pencereye geçiş yapması.

VTS003-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Müşteri/TC Kimlik Numarası" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Müşteri/TC Kimlik Numarası" alanına tıkladığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Müşteri/TC Kimlik Numarası" alanına tıklanması.

VTS004-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "İnternet Şifresi" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "İnternet Şifresi" alanına tıkladığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "İnternet Şifresi" alanına tıklanması.

VTs005-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanır. "Bireysel" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Giriş" butonuna tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Bireysel" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonuna tıklanması.

VTs006-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı" butonuna tıklandıktan sonra çıkan pencerede "Ticari" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Ticari" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı" butonuna tıklandıktan sonra çıkan pencerede "Ticari" butona tıklandığında doğru pencereye geçiş yapması.

VTs007-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Müşteri Numarası" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Ticari" butona tıklanır. "Ticari" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Müşteri Numarası" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Müşteri Numarası" alanına tıklanması.

VTs008-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Kullanıcı Kodu" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Ticari" butona tıklanır. "Ticari" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Kullanıcı Kodu" alanına tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Kullanıcı Kodu" alanına tıklanması.

VTS009-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "İnternet Şifresi" alanının test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Ticari" butona tıklanır. “Ticari” butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "İnternet Şifresi" alanına tıklandığı kontrol edilir.



Beklenen Sonuç: "İnternet Şubesi", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "İnternet Şifresi" alanına tıklanması.

VT010-Test Senaryo: Vakıfbank.com.tr sayfasında "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonunun test edilmesi.

Test Açıklaması: Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Ticari" butona tıklanır. "Ticari" butonuna tıklandıktan sonra doğru pencereye geçiş yapıp yapmadığı kontrol edilerek çıkan pencerede "Giriş" butonuna tıklandığı kontrol edilir.

Beklenen Sonuç: "İnternet Bankacılığı", "Ticari" butonuna tıklandıktan sonra çıkan pencerede "Giriş" butonuna tıklanması.

Vakıfbank Hesaplama Aracı Modülü

VT011-Test Senaryo: Vakıfbank.com.tr sayfasında ana menüdeki "Hesaplama Araçları" linkinin test edilmesi.

Test Açıklaması: Ana menüdeki "Hesaplama Araçları" linkinin üstüne gelinir. Daha sonra linke tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

Beklenen Sonuç: "Hesaplama Araçları" linkine tıklandığında doğru pencereye geçmesi.

Vakıfbank Mevduat Hesaplama Modülü

VT012-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" yazısının üstüne gelinerek, Mevduat Hesaplama alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama " alanının olması.

VT013-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Tutar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Tutar" alanına tıklanarak veri girilmesi.

VTs014-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Vade(Gün) Sayısı" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade(Gün) Sayısı" alanına tıklanarak veri girişı yapılip yapılamadıđı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Vade(Gün) Sayısı" alanına tıklanarak veri girilmesi.

VTs015-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Para Birimi" alanının test edilmesi. **Test Açıklaması:** "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Para Birimi" combobox'ından veri seçimi yapılip yapılamadıđı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Para Birimi" combobox'ından veri seçilebilmesi.

VTs016-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Vade Bitiş Tarihi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade Bitiş Tarihi" alanına tıklanarak veri girişı yapılip yapılamadıđı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Vade Bitiş Tarihi" alanına tıklanarak veri girilmesi.

VTs017-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Hesapla” butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsü açıldıktan sonra “Hesapla” butonuna tıklanması.

VT018-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Mevduat Vade Gün Sayısına göre hesapla işleminin yapılması

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Tutar”, "Vade (Gün) Sayısı" ve “Para Birimi” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapılıp yapılamadığı kontrol edilir. Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra “Tutar”, "Vade(Gün) Sayısı" ve “Para Birimi” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklayarak hesaplama işleminin gerçekleşmesi. **VT019-Test Senaryo:** Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Mevduat Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Mevduat Vade Bitiş Tarihine göre hesapla işleminin yapılması

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Tutar”, “Para Birimi” ve “Vade Bitiş Tarihi” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapılıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Mevduat Hesaplama" menüsüne tıklandıktan sonra “Tutar”, “Para Birimi” ve “Vade Bitiş Tarihi” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Vakıfbank Kredi Taksitleri Modülü

VT020-Test Senaryo: Vakıfbank.com.tr sayfasında “Hesaplama Araçları” linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede “Kredi Taksitleri” yazısının üstüne gelinerek, Kredi Taksitleri alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" alanının olması.

VTs021-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Kredi Taksitleri" butonuna tıklandıktan sonra çıkan pencerede "Kredi" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi" combobox'ından veri seçimi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Taksitleri" menüsüne tıklandıktan sonra çıkan pencerede "Kredi" combobox'ından veri seçilebilmesi.

VTs022-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Kredi Taksitleri" butonuna tıklandıktan sonra çıkan pencerede "Vade" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Taksitleri" menüsüne tıklandıktan sonra çıkan pencerede "Vade" alanına tıklanarak veri girilmesi.

VTs023-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Kredi Taksitleri" butonuna tıklandıktan sonra çıkan pencerede "Tutar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Taksitleri" menüsüne tıklandıktan sonra çıkan pencerede "Tutar" alanına tıklanarak veri girilmesi.

VTs024-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Kredi Taksitleri" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Hesapla" butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Kredi Taksitleri" menüsü açıldıktan sonra "Hesapla" butonuna tıklanması.

VT025-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Kredi Taksitleri" butonuna tıklandıktan sonra çıkan pencerede Kredi Hesaplama işleminin yapılması

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi", "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Kredi Taksitleri" menüsüne tıklandıktan sonra "Kredi", "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Vakıfbank Nakite Taksit Hesaplama Modülü

VT026-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" yazısının üstüne gelinerek, Nakite Taksit Hesaplama alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" alanının olması.

VT027-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Nakite Taksit Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Vade" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Nakite Taksit Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Vade" alanına tıklanarak veri girilmesi.

VT028-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Nakite Taksit Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Tutar" alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar" alanına tıklanarak veri girişi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Nakite Taksit Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede "Tutar" alanına tıklanarak veri girilmesi.

VT029-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Nakite Taksit Hesaplama" butonuna tıklandıktan sonra çıkan pencerede "Hesapla" butonunun test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Hesapla" butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Nakite Taksit Hesaplama" menüsü açıldıktan sonra "Hesapla" butonuna tıklanması.

VT030-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Nakite Taksit Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Nakite Taksit Hesaplama işleminin yapılması

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Nakite Taksit Hesaplama" menüsüne tıklandıktan sonra "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

Vakıfbank Tahvil Bono Hesaplama Modülü

VT031-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsünün test edilmesi.

Test Açıklaması: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede “Tahvil Bono Hesaplama” yazısının üstüne gelinerek, Tahvil Bono Hesaplama alanının sayfada olup olmadığı kontrol edilir.

Beklenen Sonuç: Hesaplama Araçları linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" alanının olması.

VT032-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Tahvil Bono Hesaplama" butonuna tıklandıktan sonra çıkan pencerede “Ürün Adı” alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Ürün Adı” combobox’ından veri seçimi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Tahvil Bono Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede “Ürün Adı” combobox’ından veri seçilebilmesi.

VT033-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Tahvil Bono Hesaplama" butonuna tıklandıktan sonra çıkan pencerede “İşlem Tipi” alanının test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “İşlem Tipi” combobox’ından veri seçimi yapıp yapılamadığı kontrol edilir.

Beklenen Sonuç: "Tahvil Bono Hesaplama" menüsüne tıklandıktan sonra çıkan pencerede “İşlem Tipi” combobox’ından veri seçimi yapıldığı görülür.

VT034-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Tahvil Bono Hesaplama" butonuna tıklandıktan sonra çıkan pencerede “Hesapla” butonunun test edilmesi.

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Hesapla” butonuna tıklanıp tıklanmadığı kontrol edilir.

Beklenen Sonuç: "Tahvil Bono Hesaplama" menüsü açıldıktan sonra “Hesapla” butonuna tıklanması.

VTs035-Test Senaryo: Vakıfbank.com.tr sayfasında "Hesaplama Araçları", "Tahvil Bono Hesaplama" butonuna tıklandıktan sonra çıkan pencerede Tahvil Bono Hesaplama işleminin yapılması

Test Açıklaması: "Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede “Ürün Adı” ve “İşlem Tipi” alanlarına veri girişı yapılp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapılp yapılamadıđı kontrol edilir.

Beklenen Sonuç: "Tahvil Bono Hesaplama" menüsüne tıklandıktan sonra “Ürün Adı” ve “İşlem Tipi” alanlarına veri girişı yapıp, “Hesapla” butonuna tıklayarak hesaplama işleminin gerçekleşmesi.

c. TCMB Uygulama Arayüzü Test Senaryoları

TCMB.gov.tr uygulama arayüzünde bazı fonksiyonel özellikler belirlenip test senaryoları yazılmıştır. TCMB uygulamasında oluşturulan test senaryoları tek başlık altında toplanmıştır. TCMB uygulama arayüzüne ait test senaryoları TTS kodu ile başlamaktadır. Mesela TTS001 olarak gösterilen test senaryosu 1 numaralı test senaryosunu ifade etmektedir.

TCMB ile Halkbank Döviz Karşılaştırma Modülü

TTS001-Test Senaryo: TCMB ana sayfasında USD kur bilgisinin alınması

Test Açıklaması: TCMB.gov.tr ana sayfasına girilir. Bu sayfada Döviz Kurları menüsünden USD Alış ve Satış bilgisinin doğru olup olmadığı kontrol edilir.

Beklenen Sonuç: TCMB.gov.tr ana sayfasına girildikten sonra Döviz Kurları menüsünden USD Alış ve Satış bilgisinin alınması.

TTS002-Test Senaryo: TCMB ana sayfasında EURO kur bilgisinin alınması

Test Açıklaması: TCMB.gov.tr ana sayfasına girilir. Bu sayfada Döviz Kurları menüsünden EURO Alış ve Satış bilgisinin doğru olup olmadığı kontrol edilir.

Beklenen Sonuç: TCMB.gov.tr ana sayfasına girildikten sonra Döviz Kurları menüsünden EURO Alış ve Satış bilgisinin alınması.

TTS003-Test Senaryo: Halkbank ana sayfasında USD kur bilgisinin alınması

Test Açıklaması: Halkbank.com.tr ana sayfasına girilir. Bu sayfada Döviz Kurları menüsünden USD Alış ve Satış bilgisinin doğru olup olmadığı kontrol edilir.

Beklenen Sonuç: Halkbank.com.tr ana sayfasına girildikten sonra Döviz Kurları menüsünden USD Alış ve Satış bilgisinin alınması.

TTS004-Test Senaryo: Halkbank ana sayfasında EURO kur bilgisinin alınması

Test Açıklaması: Halkbank.com.tr ana sayfasına girilir. Bu sayfada Döviz Kurları menüsünden EURO Alış ve Satış bilgisinin doğru olup olmadığı kontrol edilir.

Beklenen Sonuç: Halkbank.com.tr ana sayfasına girildikten sonra Döviz Kurları menüsünden EURO Alış ve Satış bilgisinin alınması.

TTS005-Test Senaryo: TCMB ve Halkbank ana sayfasından USD ve EURO kur bilgisinin alınarak karşılaştırılması

Test Açıklaması: TCMB.gov.tr ve Halkbank.com.tr ana sayfalarına girilerek, USD Alış, Satış ile EURO Alış, Satış bilgileri alınır. İki sitenin USD ve Euro bilgileri karşılaştırılarak aradaki fark %10' ndan büyükse uyarı verilmesinin sağlanması kontrol edilir. **Beklenen Sonuç:** TCMB.gov.tr ve Halkbank.com.tr sayfalarından USD Alış, Satış ile EURO USD ve Euro bilgileri karşılaştırılarak aradaki fark %10' ndan büyükse uyarı verilmesi.

2.1.3 Manuel Yazılım Test Süreci

Bu bölümde test senaryo oluşturma sürecinde hazırlanan test senaryolarının manuel test süreçleri yürütülmüştür. Manuel test sürecinde öncelikle Halkbank uygulama arayüzünün test senaryolarının manuel testleri gerçekleştirilmiştir. Sonrasında ise Vakıfbank uygulama arayüzünün testleri gerçekleştirilmiştir. Bu testler tamamlandıktan sonra TCMB uygulama arayüzüne ait test senaryolarının manuel test süreçleri gerçekleştirilmiştir.

2.1.4 Yazılım Test Otomasyon Süreci

Bu bölümde test senaryo oluşturma sürecinde hazırlanan test senaryoları Selenium Web Driver test aracı kullanarak C# programlama dilinde kodlanarak otomatize edilmiştir. Ayrıca Selenium IDE test aracı kullanılarak yine aynı test senaryoları otomatize edilmiştir. Otomatize edilen test senaryoları uygulama bazlı (Halkbank, Vakıfbank,

TCMB) olarak gruplanmıştır. Her uygulamaya özel hazırlanan örnek test otomasyon caseleri sırasıyla belirtilmiştir.

i. Halkbank Uygulamasının Otomasyon Testleri

Bu bölümde Halkbank.com.tr uygulamasına ait bazı özelliklerin test senaryoları otomatize edilmiştir.

Halkbank uygulamasının her modülü için otomatize edilen örnek test otomasyon senaryoları sırasıyla aşağıda belirtilmiştir.

Halkbank İnternet Şubesi Modülü Örnek Test Otomasyon Senaryosu

1. HTS006 numaralı senaryonun otomasyon örneği

Ana menüdeki "İnternet Şubesi" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Kurumsal" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

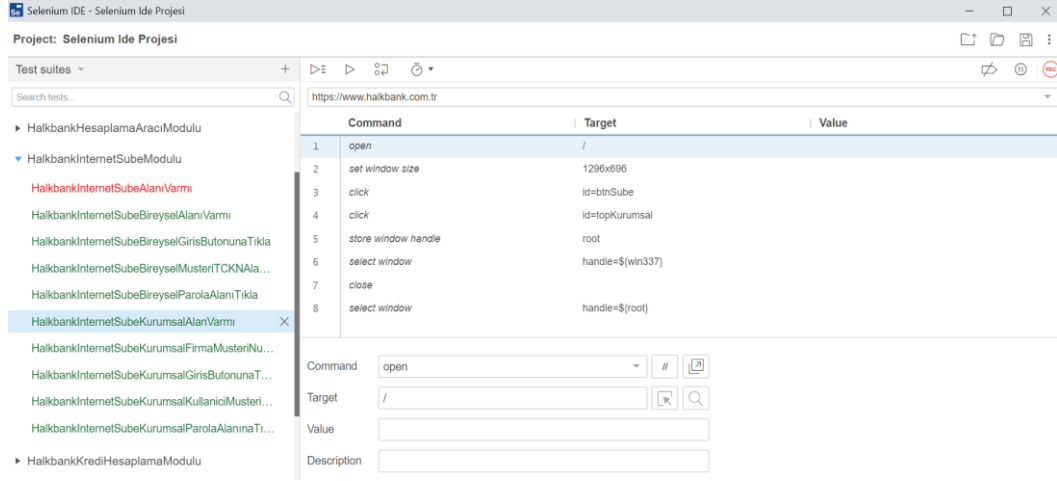
a. Selenium Web Driver kullanılarak HTS006 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankInternetSubeKurumsalAlanVarımı()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.Id("btnSube")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("topKurumsal")).Click();

    if (driver.FindElement(By.Id("topKurumsal")).Text == "Kurumsal")
    {
        Console.WriteLine("Kurumsal butonu yüklendi");
    }
    else
    {
        Console.WriteLine("HATA : Kurumsal butonu yüklenemedi");
    }

    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak HTS006 numaralı otomasyon senaryosunun gerçekleştirilmesi



Halkbank Hesaplama Aracı Modülü Örnek Test Otomasyon Senaryosu

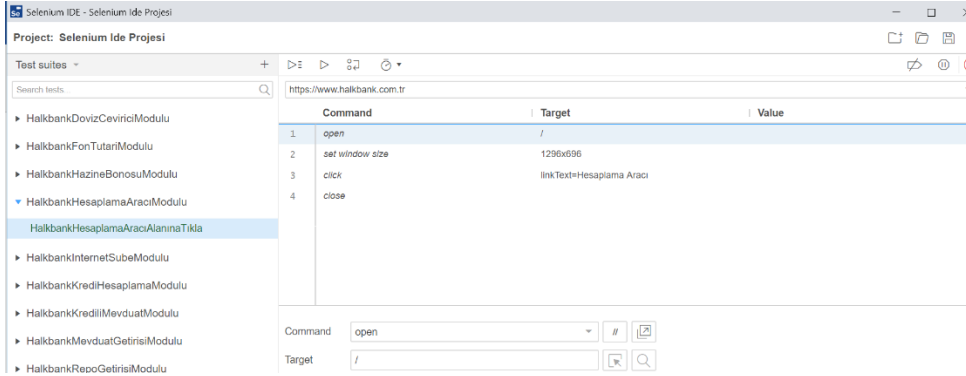
2. HTS011 numaralı senaryonun otomasyon örneği

Ana menüdeki "Hesaplama Aracı" linkinin üstüne gelinir. Daha sonra linke tıklanarak doğru pencereye geçiş yapıp yapılmadığı kontrol edilir.

a. Selenium Web Driver kullanılarak HTS011 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestClass]
0 references
public class HalkbankHesaplamaAraciModulu
{
    [TestMethod]
    0 references
    public void HalkbankHesaplamaAraciAlaninaTikla()
    {
        ChromeDriver driver = new ChromeDriver();
        driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
        Thread.Sleep(1000);
        driver.Manage().Window.Maximize();
        driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
        Console.WriteLine("HESAPLAMA ARACI tıklandı");
        Thread.Sleep(1000);
        driver.Close();
        driver.Quit();
    }
}
```

b. Selenium IDE kullanılarak HTS011 numaralı otomasyon senaryosunun gerçekleştirilmesi



Halkbank Kredi Hesaplama Modülü Örnek Test Otomasyon Senaryosu

3. HTS016 numaralı senaryonun otomasyon örneği

Halkbank.com.tr sayfasında "Kredi Hesaplama" menüsünde, "Kredi Tutarı" ve "Ay" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılmadığı kontrol edilir.

a. Selenium Web Driver kullanılarak HTS016 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankKrediHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.Id("Amount_MonthAmount")).SendKeys("11.000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ddlCreditTerm")).SendKeys("22");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("yenidenAc")).Click();
    var item1 = driver.FindElement(By.Id("lblInstallmentAmountText"));
    var Text1 = item1.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    var item2 = driver.FindElement(By.Id("lblFaizOraniText"));
    var Text2 = item2.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    var item3 = driver.FindElement(By.Id("lblTotalAmountText"));
    var Text3 = item3.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    if (Text1[0] != "" && Text2[0] != "" && Text3[0] != "")
    {
        Console.WriteLine("Kredi Hesaplandı");
    }
    else
    {
        Console.WriteLine("HATA : Kredi Hesaplanamadı");
    }
    Thread.Sleep(1000);
    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak HTS016 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1	open	/
2	set window size	1296x696
3	click	css= row:nth-child(3) > .col-md-6:nth-child(2)
4	type	id=Amount_MonthAmount 10.000
5	type	id=Amount_MonthAmount 10.000
6	click	id=ddlCreditTerm
7	select	id=ddlCreditTerm label=25 AY
8	select	id=ddlCreditTerm label=25 AY
9	click	id=ddlCreditTerm
10	mouse down at	id=yenidenAc 49,25,26,999595892004395
11	click	id=yenidenAc

Command: click
Target: id=yenidenAc
Value:
Description:

Halkbank Kredili Mevduat Modülü Örnek Test Otomasyon Senaryosu

4. HTS025 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Kredili Mevduat" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi Tutarı", "Kullanım Tarihi" ve "Ödeme Tarihi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılmadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak HTS025 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankKrediMevduatHesapla()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#content > div > div.col-md-5.hesaplamaIcon > a:nth-child(2) > span")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Amount")).SendKeys("15.000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("UseDate")).SendKeys("30.04.2020");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("PaymentDate")).SendKeys("30.12.2020");
    Thread.Sleep(1000);
    driver.FindElement(By.ClassName("btn anim fontW600 fontSize16 bgBlue1 colorWhite")).Click();
    Thread.Sleep(1000);
    var item = driver.FindElement(By.Id("FaizTotal")).Text;

    if (item != "0,00")
    {
        Console.WriteLine("Kredi Mevzuat hesaplanıldı");
    }
    else
    {
        Console.WriteLine("HATA : Kredi Mevzuat hesaplanmadı!!!");
    }
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak HTS025 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1. open	/	
2. set window size	1296x696	
3. click	linkText=Hesaplama Aracı	
4. click	css=a:nth-child(2) > span	
5. click	id=Amount	
6. type	id=Amount	15.000
7. click	id=UseDate	
8. click	css=dp_current	
9. click	id=PaymentDate	
10. click	css=dp_hover	
11. click	css=dp_hover	

Halkbank Mevduat Getirisi Modülü Örnek Test Otomasyon Senaryosu

5. HTS031 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Mevduat Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar", "Vade(Gün)" ve "TL"

alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak HTS031 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankMevduatGetirisiniHesapla()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Mevduat Getirisi")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Amount")).SendKeys("15.000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Term")).SendKeys("35");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("btnCalculateOverdraft")).Click();
    Thread.Sleep(1000);
    var tableOranText = driver.FindElement(By.ClassName("tableOran"));
    var tableOranList = tableOranText.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    if (tableOranList.Length > 1)
    {
        if (tableOranList[0] != "" && tableOranList[1] != "")
        {
            Console.WriteLine("Mevduat getirisi hesaplatıldı");
        }
        else
        {
            Console.WriteLine("HATA : Mevduat getirisi hesaplanmadı!!!");
        }
    }
    Thread.Sleep(1000);
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak HTS031 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
open	https://www.halkbank.com.tr	
set window size	1936x1056	
click	linkText=Hesaplama Aracı	
click	linkText=Mevduat Getirisi	
click	id=Amount	
type	id=Amount	5.000
click	id=CurrencyCode	
select	id=CurrencyCode	label=USD
click	id=CurrencyCode	
click	id=Term	
type	id=Term	30
click	id=btnCalculateOverdraft	

Halkbank Repo Getirisi Modülü Örnek Test Otomasyon Senaryosu

6. HTS036 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Repo Getirisi" menüsüne tıklanır. Daha sonra çıkan pencerede “Repo Vadesi” ve "Tutar” alanlarına veri girişi yapıp, “Hesapla” butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak HTS036 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankRepoGetirisiHesapla()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Repo Getirisi")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Term")).SendKeys("31");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Amount")).SendKeys("50000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("btnCalculateRepo")).Click();
    Thread.Sleep(1000);
    var item1 = driver.FindElement(By.CssSelector("#RepoForm > div > div.col.col-md-5.col-xs-offset-1 > d
    var item2 = driver.FindElement(By.CssSelector("#RepoForm > div > div:nth-child(7) > div > h2"));
    if (item1 != "0,00" && item2 != "0,00")
    { Console.WriteLine("Repo Getirisi hesaplandı"); }
    else
    { Console.WriteLine("HATA : Repo Getirisi hesaplanmadı!!!"); }
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak HTS036 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1	open	/
2	set window size	1936x1056
3	click	linkText=Hesaplama Aracı
4	click	css=^nth-child(4) > span
5	click	id=Term
6	select	id=Term label=31
7	click	id=Term
8	click	id=Amount
9	type	id=Amount 50.000
10	click	id=btnCalculateRepo
11	close	

Halkbank Hazine Bonusu Modülü Örnek Test Otomasyon Senaryosu

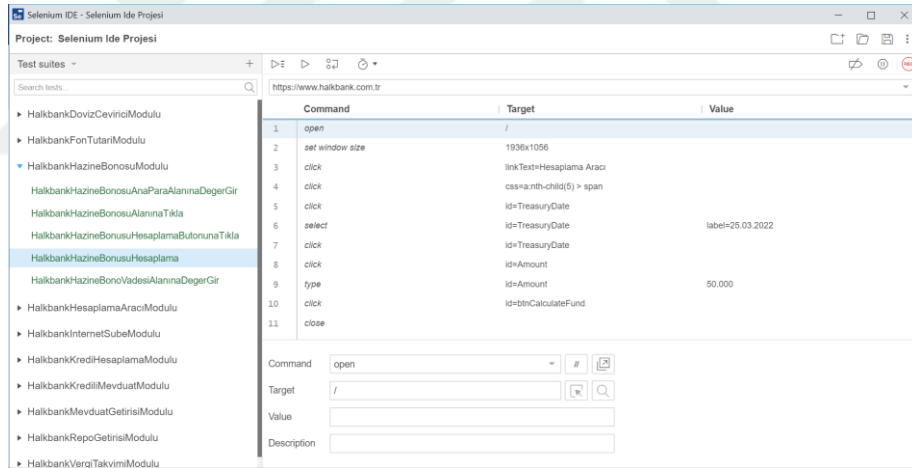
7. HTS041 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonusu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Bono Vadesi" ve "Ana Para" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

a. Selenium Web Driver kullanılarak HTS041 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankHazineBonusuHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Hazine Bonusu Hesaplama")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("TreasuryDate")).SendKeys("12.11.2021");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("Amount")).SendKeys("500000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("btnCalculateFund")).Click();
    Thread.Sleep(1000);
    var item1 = driver.FindElement(By.Id("ToplamOdenecekTutar")).Text;
    var item2 = driver.FindElement(By.Id("Nominal")).Text;
    if (item1 != "0,00" && item2 != "0,00")
    { Console.WriteLine("Hazine Bonusu hesaplanıldı"); }
    else
    { Console.WriteLine("HATA : Hazine Bonusu hesaplanmadı!!!"); }
    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak HTS041 numaralı otomasyon senaryosunun gerçekleştirilmesi



Halkbank Döviz Çevirici Modülü Örnek Test Otomasyon Senaryosu

8. HTS047 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Hazine Bonusu Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Bono Vadesi" ve "Ana Para" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılmadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak HTS047 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankDovizCevircisiHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Döviz Çevirici")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("SourceCurrency")).SendKeys("Amerikan Doları");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("SourceAmount")).SendKeys("25000");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("DestinationCurrency")).SendKeys("Euro");
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#CurrencyForm > div > div.col.col-md-12.b1 > input")).Click();
    //driver.FindElement(By.ClassName("btn anim fontw600 fontsize16 bgBlue1 colorWhite")).Click();
    if (driver.FindElement(By.Id("DestinationAmount")).Text != "")
    {
        Console.WriteLine("Döviz Çevirici hesaplandı");
    }
    else
    {
        Console.WriteLine("HATA : Döviz Çevirici hesaplanamadı");
    }
    Thread.Sleep(1000);
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak HTS047 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
3	click	linkText=Hesaplama Aracı
4	click	css=anth-child(6) > span
5	click	id=SourceCurrency
6	select	id=SourceCurrency label=Amerikan Doları
7	click	id=SourceCurrency
8	click	id=DestinationCurrency
9	select	id=DestinationCurrency label=İngiliz Sterlini
10	click	id=DestinationCurrency
11	click	id=SourceAmount
12	type	id=SourceAmount 5.000
13	click	css=.btn

Command: open
Target: /
Value:
Description:

Halkbank Vergi Takvimi Modülü Örnek Test Otomasyon Senaryosu

9. HTS048 numaralı senaryonun otomasyon örneği

Hesaplama Aracı linkine tıklandıktan sonra çıkan pencerede "Vergi Takvimi" menüsüne tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

a. Selenium Web Driver kullanılarak HTS048 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankVergiTakvimineTikla()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Vergi Takvimi")).Click();
    if (driver.FindElement(By.ClassName("hesaplamaToolHead")).Text == "Vergi Takvimi")
    {
        Console.WriteLine("Vergi Takvimi alanı yüklendi");
    }
    else
    {
        Console.WriteLine("HATA : Vergi Takvimi alanı yüklenemedi");
    }

    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak HTS048 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1 open	/	
2 set window size	1936x1056	
3 click	linkText=Hesaplama Aracı	
4 click	css=anth-child(7) > span	
5 close		

Command: open
Target: /
Value:
Description:

Halkbank Fon Tutarı Modülü Örnek Test Otomasyon Senaryosu

10. HTS053 numaralı senaryonun otomasyon örneği

"Hesaplama Aracı" linkine tıklandıktan sonra çıkan pencerede "Fon Tutarı Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Yatırım Fonu" ve "Pay Adedi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

a. Selenium Web Driver kullanılarak HTS053 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void HalkbankFonTutarıHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Fon Tutarı Hesaplama")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("FundType")).SendKeys("Para Piyasası Fonu");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("NumberOfShares")).SendKeys("5");
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#FundForm > div > div:nth-child(5) > input")).Click();
    //driver.FindElement(By.ClassName("btn anim fontW600 fontSize16 bgBlue1 colorWhite")).Click();
    Thread.Sleep(1000);
    if (driver.FindElement(By.CssSelector("#FundForm > div > div.col.col-md-6.col-xs-offset-3 > div > h2")).Text != "0,00")
    {
        Console.WriteLine("Fon tutarı hesaplandı");
    }
    else
    {
        Console.WriteLine("HATA : Fon tutarı hesaplanamadı");
    }
    Thread.Sleep(1000);
    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak HTS053 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1	open	/
2	set window size	1936x1056
3	click	linkText=Hesaplama Aracı
4	click	css=az:nth-child(8) > span
5	click	id=FundType
6	select	id=FundType label=Para Piyasası Fonu
7	click	id=FundType
8	click	id=NumberOfShares
9	type	id=NumberOfShares 51
10	click	css=.btn
11	close	

Command: open
Target: /
Value:
Description:

ii. Vakıfbank Uygulamasının Otomasyon Testleri

Bu bölümde Vakıfbank.com.tr uygulamasına ait bazı özelliklerin test senaryoları otomatize edilmiştir.

Vakıfbank uygulamasının her modülü için otomatize edilen örnek test otomasyon senaryoları sırasıyla aşağıda belirtilmiştir.

Vakıfbank İnternet Bankacılığı Modülü Örnek Test Otomasyon Senaryosu

1. VTS002 numaralı senaryonun otomasyon örneği

Ana menüdeki "İnternet Bankacılığı" butonunun üstüne gelinir. Daha sonra çıkan pencerede "Bireysel" butona tıklanarak doğru pencereye geçiş yapıp yapmadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak VTS002 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void VakıfbankİnternetBankacılığıBireyselAlanıVarmı()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.ClassName("internetBank")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.LinkText("Bireysel")).Click();
    Thread.Sleep(1000);
    if (driver.FindElement(By.LinkText("Bireysel")).Text == "Bireysel")
    {
        Console.WriteLine("Bireysel butonu yüklendi");
    }
    else
    {
        Console.WriteLine("HATA : Bireysel butonu yüklenemedi");
    }
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanarak VTS002 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1. open	https://www.vakifbank.com.tr/	
2. set window size	1296x696	
3. click	linkText=Bireysel	
4. store window handle	root	
5. select window	handle=\${win5433}	
6. close		
7. select window	handle=\${root}	
8. close		

Command: open
Target: https://www.vakifbank.com.tr/
Value:
Description:

Vakıfbank Mevduat Hesaplama Modülü Örnek Test Otomasyon Senaryosu

2. VTS018 numaralı senaryonun otomasyon örneği

"Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Mevduat Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Tutar", "Vade(Gün) Sayısı" ve "Para Birimi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak VTS018 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void VakifbankMevduatVadeGunSayısınaGoreHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/mevduat-hesaplama-araci.aspx?pageID=2376");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    //driver.FindElement(By.Id("select2-ctl100_ctl100_ddlParaBirimi-container")).SendKeys("USD");
    driver.FindElement(By.CssSelector("#select2-ctl100_ctl100_ddlParaBirimi-container")).SendKeys("USD");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ctl100_ctl100_txtMevduatTutar")).SendKeys("500000");
    Thread.Sleep(1000);
    //driver.FindElement(By.Id("mevduatVadeGun")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ctl100_ctl100_txtVadeGunSayisi")).SendKeys("30");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ctl100_ctl100_btnHesapla")).Click();
    Thread.Sleep(1000);
    if (driver.FindElement(By.CssSelector("#Form1 > div.container-fluid.content > div:nth-child(3) > div > div:nth-child(1)")).Text.Contains("Mevduat Vade Gün Sayısına Göre hesaplandı"))
    {
        Console.WriteLine("Mevduat Vade Gün Sayısına Göre hesaplandı");
    }
    else
    {
        Console.WriteLine("HATA : Mevduat Vade Gün Sayısına Göre hesaplanamadı");
    }
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak VTS018 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1	open	https://www.vakifbank.com.tr/
2	set window size	1296x696
3	click	linkText=Hesaplama Araçları
4	click	css= tool-1 h2
5	click	id=select2-ctl100_ctl100_ddlParaBirimi-container
6	click	id=ctl100_ctl100_txtMevduatTutar
7	type	id=ctl100_ctl100_txtMevduatTutar 50000
8	click	id=ctl100_ctl100_btnVadeGunSayisi
9	type	id=ctl100_ctl100_btnVadeGunSayisi 30
10	click	id=ctl100_ctl100_btnHesapla
11	close	

Command: open
Target: https://www.vakifbank.com.tr/
Value:
Description:

Vakıfbank Kredi Taksitleri Modülü Örnek Test Otomasyon Senaryosu

3. VTS025 numaralı senaryonun otomasyon örneği

"Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Kredi Taksitleri" menüsüne tıklanır. Daha sonra çıkan pencerede "Kredi", "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılamadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak VTS025 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void VakıfbankKrediHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/kredi-hesaplama-araci.aspx?pageID=1333");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    SelectElement secim = new SelectElement(driver.FindElement(By.CssSelector("#select2-ctl100_ctl110_ctl100_ddlKredi-container")));
    //secim.SelectByText("TAKSİTLİ EK HESAP");
    secim.SelectByIndex(3);
    // driver.FindElement(By.CssSelector("#select2-ctl100_ctl110_ctl100_ddlKredi-container")).Click();
    // driver.FindElement(By.CssSelector("#select2-ctl100_ctl110_ctl100_ddlKredi-container")).SendKeys("TAKSİTLİ EK HESAP");
    //driver.FindElement(By.Id("select2-ctl100_ctl110_ctl100_ddlKredi-container")).SendKeys("TAKSİTLİ EK HESAP");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ctl100_ctl110_ctl100_txtVade")).SendKeys("36");
    Thread.Sleep(1000);
    driver.FindElement(By.Id("ctl100_ctl110_ctl100_txtTutar")).SendKeys("25000");
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ctl100_ctl110_ctl100_btnHesapla"));
    //driver.FindElement(By.Id("ctl100_ctl110_ctl100_btnHesapla")).Click();
    if (driver.FindElement(By.CssSelector("#ctl100_ctl110_ctl100_ajaxPanel > h2")).Text == "Kredi Bilgileri")
    {
        Console.WriteLine("Kredi hesaplandı");
    }
    else
    {
        Console.WriteLine("HATA : Kredi hesaplanamadı");
    }
    driver.Close();
    driver.Quit();
}
```

- b. Selenium IDE kullanılarak VTS025 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1. open	https://www.vakifbank.com.tr/	
2. set window size		1296x696
3. click	linkText=Hesaplama Araçları	
4. click	css= col-xs-12:nth-child(2) h2	
5. click	id=select2-ctl100_ctl110_ctl100_ddlKredi-container	TAKSİTLİ EK HESAP
6. click	id=ctl100_ctl110_ctl100_txtVade	
7. type	id=ctl100_ctl110_ctl100_txtVade	10
8. click	id=ctl100_ctl110_ctl100_txtTutar	
9. type	id=ctl100_ctl110_ctl100_txtTutar	50000
10. click	css= col-sm-6:nth-child(1)	
11. click	id=ctl100_ctl110_ctl100_btnHesapla	

Command: close
Target:
Value:
Description:

Vakıfbank Nakite Taksit Hesaplama Modülü Örnek Test Otomasyon Senaryosu

4. VTS030 numaralı senaryonun otomasyon örneği

"Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Nakite Taksit Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Vade" ve "Tutar" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılmadığı kontrol edilir.

- a. Selenium Web Driver kullanılarak VTS030 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void VakifbankNakiteTaksitHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/nakite-taksit-hesaplama-araci.aspx?pageID=2377");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.CssSelector("#ctl00_ctl10_ctl00_txtVade")).Clear();
    //driver.FindElement(By.Id("sctl00_ctl10_ctl00_txtVade")).Clear();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ctl00_ctl10_ctl00_txtVade")).SendKeys("12");
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ctl00_ctl10_ctl00_txtTutar")).Clear();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ctl00_ctl10_ctl00_txtTutar")).SendKeys("500");
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ctl00_ctl10_ctl00_btnHesapla")).Click();
    //driver.FindElement(By.Id("ctl00_ctl10_ctl00_btnHesapla")).Click();
    Thread.Sleep(1000);
    var tableOranText = driver.FindElement(By.CssSelector("#Form1 > div.container-fluid.content > div:nth-child(3) > div > div:nth-child(1)"));
    var tableOranList = tableOranText.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    //Console.WriteLine("Length " + tableOranList.Length);
    if (tableOranList.Length > 1)
    {
        if (tableOranList[0] != "" && tableOranList[1] != "")
            Console.WriteLine("Nakite taksit hesaplatıldı");
    }
    else
    {
        Console.WriteLine("HATA : Nakite taksit hesaplanmadı!!!");
    }
    driver.Close();
}
```

- b. Selenium IDE kullanılarak VTS030 numaralı otomasyon senaryosunun gerçekleştirilmesi

Command	Target	Value
1. open	https://www.vakifbank.com.tr/	
2. set window size	1296x696	
3. mouse over	linkText=Ürün ve Hizmet Ücretleri	
4. run script	window.scrollTo(0,400)	
5. click	linkText=Hesaplama Araçları	
6. click	css=col-xs-12:nth-child(3) h2	
7. click	css=container-fluid > .row:nth-child(3)	
8. type	id=ctl00_ctl10_ctl00_txtVade	11
9. click	css=container-fluid > .row:nth-child(3)	
10. type	id=ctl00_ctl10_ctl00_txtTutar	550
11. click	id=ctl00_ctl10_ctl00_btnHesapla	

Vakıfbank Tahvil Bono Hesaplama Modülü Örnek Test Otomasyon Senaryosu

5. VTS035 numaralı senaryonun otomasyon örneği

"Hesaplama Araçları" linkine tıklandıktan sonra çıkan pencerede "Tahvil Bono Hesaplama" menüsüne tıklanır. Daha sonra çıkan pencerede "Ürün Adı" ve "İşlem Tipi" alanlarına veri girişi yapıp, "Hesapla" butonuna tıklanarak hesaplama işleminin yapıp yapılmadığı kontrol edilir.

a. Selenium Web Driver kullanılarak VTS035 numaralı otomasyon senaryosunun gerçekleştirilmesi

```
[TestMethod]
public void VakıfbankTahvilBonoHesaplama()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.CssSelector("#Form1 > div.container-fluid.tabs > div > div > div > div > div > div > a.tab-icon.tab-01.selected.slick-slide.slick-current.slick-active")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#Form1 > div.container-fluid.tabContents > div > div > div:nth-child(1) > div:nth-child(2) > div:nth-child(4) > div > a > h2")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#select2-ct100_ct110_ct100_ddlTahvilBonoUrun-container")).Click();
    driver.FindElement(By.CssSelector("#select2-ct100_ct110_ct100_ddlTahvilBonoUrun-container")).SendKeys("Tahvil - Bono");
    // driver.FindElement(By.Id("select2-ct100_ct110_ct100_ddlTahvilBonoUrun-container")).Click();
    // driver.FindElement(By.Id("ct100_ct110_ct100_ddlTahvilBonoUrun")).Click();
    Thread.Sleep(1000);
    // driver.FindElement(By.LinkText("Tahvil - Bono")).Click();
    driver.FindElement(By.CssSelector("#select2-ct100_ct110_ct100_ddlTahvilIslemTipi-container")).Click();
    driver.FindElement(By.CssSelector("#select2-ct100_ct110_ct100_ddlTahvilIslemTipi-container")).SendKeys("Alış");
    // driver.FindElement(By.LinkText("Alış")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#ct100_ct110_ct100_btnHesapla")).Click();
    // driver.FindElement(By.Id("ct100_ct110_ct100_btnHesapla")).Click();
    Thread.Sleep(1000);
    var tableOranText = driver.FindElement(By.CssSelector("#ct100_ct110_ct100_gjxPanel > div.table-responsive > table > tbody"));
    var tableOranList = tableOranText.Text.Split(new[] { "\n\n" }, StringSplitOptions.None);
    //Console.WriteLine("Length " + tableOranList.Length);
    if (tableOranList.Length > 1)
    {
        if (tableOranList[0] != "" && tableOranList[1] != "")
        {
            Console.WriteLine("Tahvil Bono hesaplatıldı");
        }
        else
        {
            Console.WriteLine("HATA : Tahvil Bono hesaplanmadı!!!");
        }
    }
    driver.Close();
    driver.Quit();
}
```

b. Selenium IDE kullanılarak VTS035 numaralı otomasyon senaryosunun gerçekleştirilmesi

The screenshot shows the Selenium IDE interface. The top bar indicates the project is 'Selenium Ide Projesi'. The left pane lists various test cases, with 'VakıfbankTahvilBonoHesaplama' selected. The right pane displays a table of test steps for this case, with the following columns: Command, Target, and Value.

Command	Target	Value
1. open	https://www.vakifbank.com.tr/	
2. set window size	1296x696	
3. click	linkText=Hesaplama Araçları	
4. click	css= col-xs-12:nth-child(4) h2	
5. click	id=select2-ct100_ct110_ct100_ddlTahvilBonoUrun-container	
6. click	id=ct100_ct110_ct100_ddlTahvilIslemTipi	
7. select	id=ct100_ct110_ct100_ddlTahvilIslemTipi	label=Alış
8. click	id=ct100_ct110_ct100_ddlTahvilIslemTipi	
9. click	css= baslik > td:nth-child(5)	
10. close		

Below the table, there is a section for the selected step (Step 9). It shows the Command as 'click', the Target as 'css= baslik > td:nth-child(5)', and the Value as an empty field. The Description field is also empty.

iii. TCMB Uygulamasının Otomasyon Testleri

Bu bölümde Halkbank.com.tr uygulamasına ait bazı özelliklerin test senaryoları otomatize edilmiştir.

TCMB uygulamasının her modülü için otomatize edilen örnek test otomasyon senaryoları sırasıyla aşağıda belirtilmiştir.

TCMB ile Halkbank Döviz Karşılaştırma Modülü Örnek Test Otomasyon Senaryosu

TCMB.gov.tr ve Halkbank.com.tr ana sayfalarına girilerek, USD Alış, Satış ile EURO Alış, Satış bilgileri alınır. İki sitenin USD ve Euro bilgileri karşılaştırılarak aradaki fark %10' ndan büyükse uyarı verilmesinin sağlanması

```
[TestMethod]
public void TCMBHalkbankKurKarsilastirma()
{
    //ChromeDriver driver = new ChromeDriver();
    InternetExplorerDriver driver = new InternetExplorerDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    driver.Manage().Window.Maximize();

    /* Console.WriteLine(driver.FindElement(By.Id("tab_ic_piyasa")).Text);

    Console.WriteLine("HALKBANK KURLAR alındı");*/
    Thread.Sleep(2000);

    var item = driver.FindElement(By.Id("tab_ic_piyasa"));
    var list = item.Text.Split(new[] { "\r\n" }, StringSplitOptions.None);
    // Console.WriteLine(list[0]);
    var FXFirst = list[0].Split(new[] { " " }, StringSplitOptions.None);

    float USDBuyingHalkbank = 0;
    float USDSellingHalkbank = 0;

    if (FXFirst[0] == "USD")
    {
        USDBuyingHalkbank = float.Parse(FXFirst[1]);
        USDSellingHalkbank = float.Parse(FXFirst[2]);
        Console.WriteLine("Halkbank USD Alış : " + USDBuyingHalkbank);
        Console.WriteLine("Halkbank USD Satış : " + USDSellingHalkbank);
    }

    var FXSecond = list[1].Split(new[] { " " }, StringSplitOptions.None);
    float EURBuyingHalkbank = 0;
    float EURSellingHalkbank = 0;
```

```
if (FXSecond[0] == "EUR")
{
    EURBuyingHalkbank = float.Parse(FXSecond[1]);
    EURSellingHalkbank = float.Parse(FXSecond[2]);
    Console.WriteLine("Halkbank EUR Alış : " + EURBuyingHalkbank);
    Console.WriteLine("Halkbank EUR Satış : " + EURSellingHalkbank + "\n");
}

ChromeDriver driverTCMB = new ChromeDriver();

driverTCMB.Navigate().GoToUrl("https://www.tcmb.gov.tr/");
//driverTCMB.Manage().Window.Maximize();
var element = driverTCMB.FindElement(By.Id("euroBuying"));
Actions actions = new Actions(driverTCMB);
actions.MoveToElement(element).Perform();
Thread.Sleep(2000);

decimal d1 = decimal.Parse(driverTCMB.FindElement(By.Id("usdBuYing")).Text, CultureInfo.InvariantCulture);

decimal USDBuyingTCMB = decimal.Parse(driverTCMB.FindElement(By.Id("usdBuYing")).Text, CultureInfo.InvariantCulture);
decimal USDSellingTCMB = decimal.Parse(driverTCMB.FindElement(By.Id("usdSelling")).Text, CultureInfo.InvariantCulture);

decimal EURBuyingTCMB = decimal.Parse(driverTCMB.FindElement(By.Id("euroBuYing")).Text, CultureInfo.InvariantCulture);
decimal EURSellingTCMB = decimal.Parse(driverTCMB.FindElement(By.Id("euroSelling")).Text, CultureInfo.InvariantCulture);

Console.WriteLine("TCMB USD Alış : " + USDBuyingTCMB);
Console.WriteLine("TCMB USD Satış : " + USDSellingTCMB);

Console.WriteLine("TCMB EUR Alış : " + EURBuyingTCMB);
Console.WriteLine("TCMB EUR Satış : " + EURSellingTCMB + "\n");
```

```

if (USDBuyingTCMB < (decimal)USDBuyingHalkbank)
    Console.WriteLine("HALKBANK USD Alış Kuru TCMB den daha yüksek !!!!");

if (USDSELLINGTCMB < (decimal)USDSELLINGHalkbank)
    Console.WriteLine("HALKBANK USD Satış Kuru TCMB den daha yüksek !!!!");

if (EURBuyingTCMB < (decimal)EURBuyingHalkbank)
    Console.WriteLine("HALKBANK EUR Alış Kuru TCMB den daha yüksek !!!!");

if (EURSELLINGTCMB < (decimal)EURSELLINGHalkbank)
    Console.WriteLine("HALKBANK EUR Satış Kuru TCMB den daha yüksek !!!!");

// Console.WriteLine("USD Alış Kuru Farkı : " + ((USDBuyingTCMB - (decimal)USDBuyingHalkbank) / (decimal)USDBuyingHalkbank));

if (((USDBuyingTCMB - (decimal)USDBuyingHalkbank) / (decimal)USDBuyingHalkbank) > (decimal)0.1)
    Console.WriteLine("HALKBANK USD Alış Kuru TCMB USD Alış Kuru arasındaki Fark %10 dan Fazla !!!!!" +
        ((USDBuyingTCMB - (decimal)USDBuyingHalkbank) / (decimal)USDBuyingHalkbank));

if (((USDSELLINGTCMB - (decimal)USDSELLINGHalkbank) / (decimal)USDSELLINGHalkbank) > (decimal)0.1)
    Console.WriteLine("HALKBANK USD Satış Kuru TCMB USD Satış Kuru arasındaki Fark %10 dan Fazla !!!!!" +
        ((USDSELLINGTCMB - (decimal)USDSELLINGHalkbank) / (decimal)USDSELLINGHalkbank));

if (((EURBuyingTCMB - (decimal)EURBuyingHalkbank) / (decimal)EURBuyingHalkbank) > (decimal)0.1)
    Console.WriteLine("HALKBANK EUR Alış Kuru TCMB EUR Alış Kuru arasındaki Fark %10 dan Fazla !!!!!" +
        ((EURBuyingTCMB - (decimal)EURBuyingHalkbank) / (decimal)EURBuyingHalkbank));

if (((EURSELLINGTCMB - (decimal)EURSELLINGHalkbank) / (decimal)EURSELLINGHalkbank) > (decimal)0.1)
    Console.WriteLine("HALKBANK EUR Satış Kuru TCMB EUR Satış Kuru arasındaki Fark %10 dan Fazla !!!!!" +
        ((EURSELLINGTCMB - (decimal)EURSELLINGHalkbank) / (decimal)EURSELLINGHalkbank));

Thread.Sleep(2000);
driver.Close();
}

```

2.1.5 Selenium Web Driver Kullanılarak Otomatik Testlerin Optimizasyonu

Bir önceki bölümde senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Bu bölümde ise tüm senaryoların eş zamanlı yürütülebilir hale getirilebilmesi için oluşturulan senaryolar optimize edilmiştir. Optimize etme işlemi sonlandıktan sonra herhangi bir modül çalıştırıldığında sırayla o modülde yer alan senaryolar otomatik olarak çalışmaktadır.

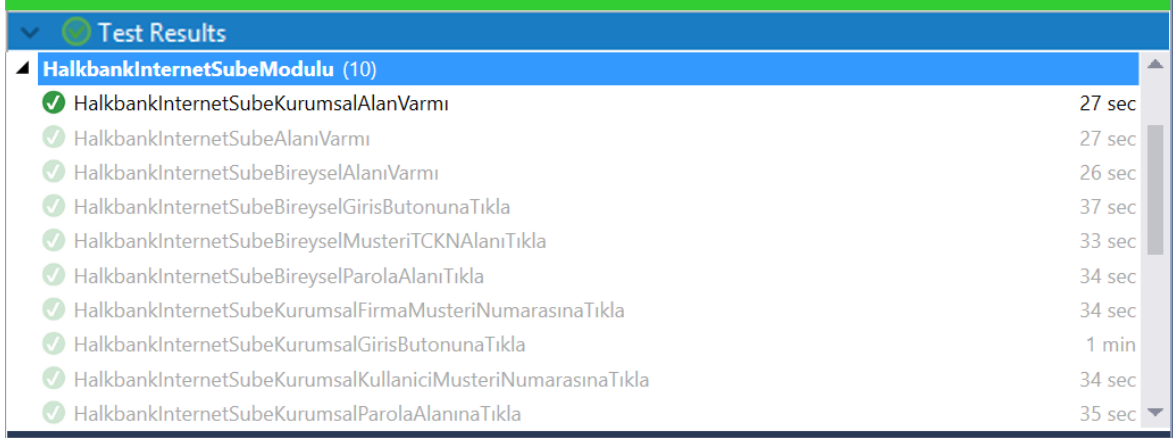
a. Halkbank Uygulamasının Testlerinin Optimizasyonu

Halkbank uygulamasının “İnternet Şubesi”, “Kredi Hesaplama”, “Hesaplama Aracı”, “Kredili Mevzuat”, “Mevduat Getirisi”, “Repo Getirisi”, “Hazine Bonosu”, “Döviz Çevirici”, “Fon Tutarı”, “Vergi Takvimi” modülleri için 53 adet otomatik test senaryosunun optimizasyon çalışması yapılmıştır. Her modül için yapılan optimizasyon işlemi sırasıyla aşağıda belirtilmiştir.

Halkbank İnternet Şubesi Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. İnternet Şubesi modülünde bu amaca yönelik kurgulanan senaryo sayısı 10 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 5 dakika 47 saniye sonunda

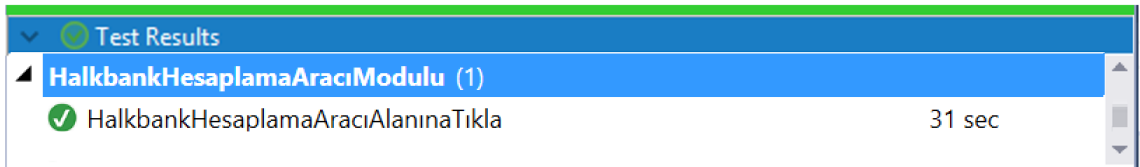
süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.



Test Results	
HalkbankInternetSubeModulu (10)	
✓ HalkbankInternetSubeKurumsalAlanVarmi	27 sec
✓ HalkbankInternetSubeAlanVarmi	27 sec
✓ HalkbankInternetSubeBireyselAlanVarmi	26 sec
✓ HalkbankInternetSubeBireyselGirisButonunaTikla	37 sec
✓ HalkbankInternetSubeBireyselMusteriTCKNAAlanıTikla	33 sec
✓ HalkbankInternetSubeBireyselParolaAlanıTikla	34 sec
✓ HalkbankInternetSubeKurumsalFirmaMusteriNumarasınaTikla	34 sec
✓ HalkbankInternetSubeKurumsalGirisButonunaTikla	1 min
✓ HalkbankInternetSubeKurumsalKullaniciMusteriNumarasınaTikla	34 sec
✓ HalkbankInternetSubeKurumsalParolaAlanınaTikla	35 sec

Halkbank Hesaplama Aracı Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Hesaplama Aracı modülünde bu amaca yönelik kurgulanan senaryo sayısı bir tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 31 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.



Test Results	
HalkbankHesaplamaAraciModulu (1)	
✓ HalkbankHesaplamaAraciAlanınaTikla	31 sec

Halkbank Kredi Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Kredi Hesaplama modülünde bu amaca yönelik kurgulanan senaryo sayısı 8 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 3 dakika 53 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
HalkbankKrediHesaplamaModulu (8)	
✓ HalkbankIhtiyacAlanınaTıkla	28 sec
✓ HalkbankKonutAlanınaTıkla	28 sec
✓ HalkbankKrediHesaplama	33 sec
✓ HalkbankKrediHesaplamaAlanıVarmı	30 sec
✓ HalkbankKrediHesaplamaAyAlanınaDegerGir	27 sec
✓ HalkbankKrediHesaplamaButonunaTıkla	30 sec
✓ HalkbankKrediTutarıAlanınaDegerGir	29 sec
✓ HalkbankTasitAlanınaTıkla	28 sec

Halkbank Kredili Mevduat Modülü

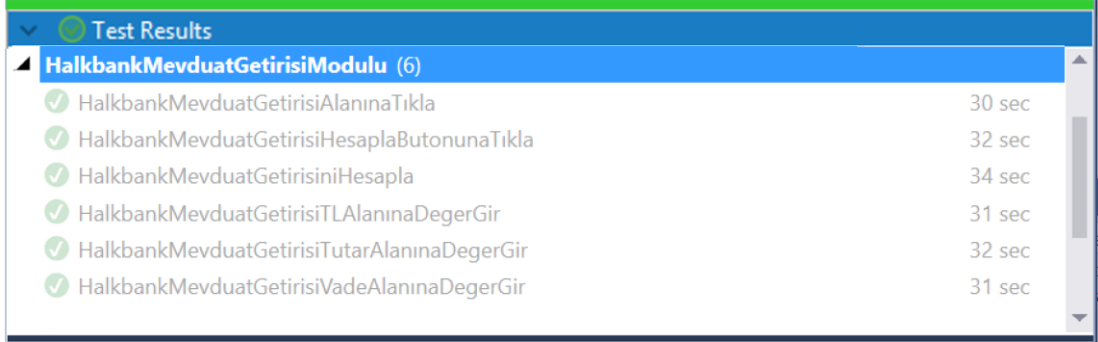
Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Kredili Mevduat modülünde bu amaca yönelik kurgulanan senaryo sayısı 6 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında 3 dakika 21 saniye sonunda süreç tamamlanmıştır. Optimizasyon işlemi sona erdikten sonra çalıştırılan otomatik testlerden 3 tanesi başarılı, diğer 3 tanesi de başarısız olarak sonuçlanmıştır. Bu modül işletilmeden önce otomatize olmayan HTS022, HTS023, HTS025 senaryolarının testleri manuel olarak gerçekleştirilmelidir. Bu durum otomatik testlerin işletilmesinde manuel testin gerçekleştirilmesini zorunlu kılmaktadır.

Test Results	
HalkbankKrediliMevduatModulu (6)	
✗ HalkbankKrediMevduatHesapla	33 sec
✗ HalkbankKullanımTarihiAlanınaDegerGir	35 sec
✗ HalkbankOdemeTarihiAlanınaDegerGir	35 sec
✓ HalkbankKrediMevduatAlanınaTıkla	31 sec
✓ HalkbankKrediMevduatHesaplaButonunaTıkla	33 sec
✓ HalkbankKrediMevduatTutarıAlanınaDegerGir	34 sec

Halkbank Mevduat Getirisi Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Mevduat Getirisi modülünde bu amaca yönelik kurgulanan senaryo sayısı 6 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 3 dakika 10 saniye

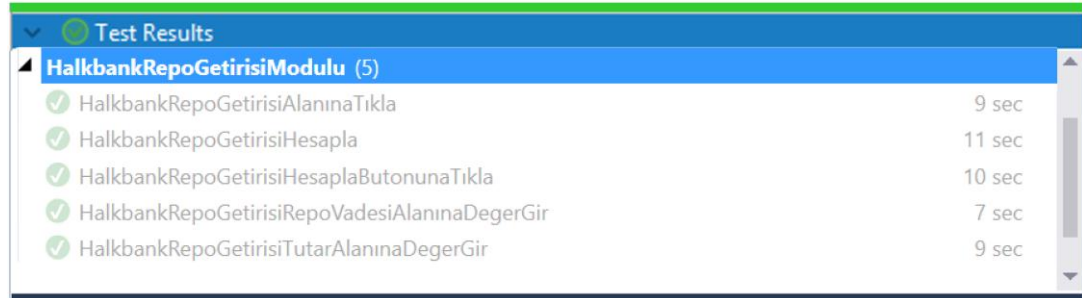
sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.



Test Results	
HalkbankMevduatGetirisiModulu (6)	
✓ HalkbankMevduatGetirisiAlanınaTıkla	30 sec
✓ HalkbankMevduatGetirisiHesaplaButonunaTıkla	32 sec
✓ HalkbankMevduatGetirisiHesapla	34 sec
✓ HalkbankMevduatGetirisiTLAlanınaDegerGir	31 sec
✓ HalkbankMevduatGetirisiTutarAlanınaDegerGir	32 sec
✓ HalkbankMevduatGetirisiVadeAlanınaDegerGir	31 sec

Halkbank Repo Getirisi Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Repo Getirisi modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 46 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.



Test Results	
HalkbankRepoGetirisiModulu (5)	
✓ HalkbankRepoGetirisiAlanınaTıkla	9 sec
✓ HalkbankRepoGetirisiHesapla	11 sec
✓ HalkbankRepoGetirisiHesaplaButonunaTıkla	10 sec
✓ HalkbankRepoGetirisiRepoVadesiAlanınaDegerGir	7 sec
✓ HalkbankRepoGetirisiTutarAlanınaDegerGir	9 sec

Halkbank Hazine Bonosu Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Hazine Bonosu modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 49 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
HalkbankHazineBonosuModulu (5)	
✓ HalkbankHazineBonosuAlanınaTıkla	9 sec
✓ HalkbankHazineBonosuAnaParaAlanınaDegerGir	9 sec
✓ HalkbankHazineBonoVadesiAlanınaDegerGir	10 sec
✓ HalkbankHazineBonusuHesaplama	11 sec
✓ HalkbankHazineBonusuHesaplamaButonunaTıkla	10 sec

Halkbank Döviz Çevirici Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Döviz Çevirici modülünde bu amaca yönelik kurgulanan senaryo sayısı 6 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 3 dakika 15 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
HalkbankDovizCeviriciModulu (6)	
✓ HalkbankDovizCeviriciAlanınaTıkla	33 sec
✓ HalkbankDovizCeviriciButonunaTıkla	31 sec
✓ HalkbankDovizCeviriciCevrilecegiParaBirimiAlanınaDegerGir	32 sec
✓ HalkbankDovizCeviriciHesaplama	35 sec
✓ HalkbankDovizCeviriciMiktarAlanınaDegerGir	31 sec
✓ HalkbankDovizCeviriciParaBirimiAlanınaDegerGir	33 sec

Halkbank Vergi Takvimi Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Vergi Takvimi modülünde bu amaca yönelik kurgulanan senaryo sayısı bir tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 29 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
HalkbankVergiTakvimiModulu (1)	
✓ HalkbankVergiTakvimineTıkla	29 sec

Halkbank Fon Tutarı Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Fon Tutarı Hesaplama modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 2 dakika 44 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.



Test Results	
HalkbankFonTutariModulu (5)	
✓ HalkbankFonTutariAlaninaTikla	32 sec
✓ HalkbankFonTutariHesaplaButonunaTikla	32 sec
✓ HalkbankFonTutariHesaplama	34 sec
✓ HalkbankPayAdediAlaninaDegerGir	32 sec
✓ HalkbankYatirimFonuAlaninaDegerGir	34 sec

b. Vakıfbank Uygulamasının Testlerinin Optimizasyonu

Vakıfbank Vakıfbank uygulamasının “İnternet Bankacılığı”, “Hesaplama Aracı”, “Mevduat Hesaplama”, “Kredi Taksitleri”, “Nakite Taksit Hesaplama”, “Tahvil Bono Hesaplama” modülleri için 35 adet otomatik test senaryosunun optimizasyon çalışması yapılmıştır. Her modül için yapılan optimizasyon işlemi sırasıyla aşağıda belirtilmiştir.

Vakıfbank İnternet Bankacılığı Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. İnternet Bankacılığı modülünde bu amaca yönelik kurgulanan senaryo sayısı 10 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 4 dakika 39 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
VakıfbankİnternetSubeModulu (10)	
✓ VakıfbankBireyselİnternetBankacılığıGirisButonunaTıkla	28 sec
✓ VakıfbankBireyselİnternetBankacılığıİnternetSifresiAlanınaTıkla	27 sec
✓ VakıfbankBireyselİnternetBankacılığıMusteriTCKNALanınaTıkla	27 sec
✓ VakıfbankİnternetBankacılığıAlanıVarmı	27 sec
✓ VakıfbankİnternetBankacılığıBireyselAlanıVarmı	30 sec
✓ VakıfbankİnternetBankacılığıTicariAlanıVarmı	31 sec
✓ VakıfbankTicariİnternetBankacılığıGirisButonunaTıkla	28 sec
✓ VakıfbankTicariİnternetBankacılığıİnternetSifresiAlanınaTıkla	27 sec
✓ VakıfbankTicariİnternetBankacılığıKullanıcıKoduAlanınaTıkla	27 sec
✓ VakıfbankTicariİnternetBankacılığıMusteriTCKNALanınaTıkla	27 sec

Vakıfbank Hesaplama Aracı Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Hesaplama Aracı modülünde bu amaca yönelik kurgulanan senaryo sayısı bir tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 33 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
VakıfbankHesaplamaAraciModulu (1)	
✓ VakıfbankHesaplamaAraciAlanınaTıkla	33 sec

Vakıfbank Mevduat Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Mevduat Hesaplama modülünde bu amaca yönelik kurgulanan senaryo sayısı 8 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında 3 dakika 33 saniye sonunda süreç tamamlanmıştır. Optimizasyon işlemi sona erdikten sonra çalıştırılan otomatik testlerden 4 tanesi başarılı, diğer 4 tanesi de başarısız olarak sonuçlanmıştır. Bu modül işletilmeden önce otomatize olmayan VTS015, VTS017, VTS018, VTS019 senaryolarının testleri manuel olarak gerçekleştirilmelidir. Bu durum otomatik testlerin işletilmesinde manuel testin gerçekleştirilmesini zorunlu kılmaktadır.

Test Results	
VakıfbankMevduatHesaplamaModulu (8)	
✗ VakıfbankMevduatHesaplamaButonunaTıkla	28 sec
✗ VakıfbankMevduatHesaplamaParaBirimiAlanınaDegerGir	27 sec
✗ VakıfbankMevduatVadeBitisTarihineGoreHesaplama	27 sec
✗ VakıfbankMevduatVadeGunSayısınaGoreHesaplama	6 sec
✓ VakıfbankMevduatHesaplamaAlanınaTıkla	31 sec
✓ VakıfbankMevduatHesaplamaTutarAlanınaDegerGir	30 sec
✓ VakıfbankMevduatHesaplamaVadeBitisTarihiAlanınaDegerGir	34 sec
✓ VakıfbankMevduatHesaplamaVadeGunSavisiAlanınaDegerGir	30 sec

Vakıfbank Kredi Taksitleri Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Kredi Taksitleri modülünde bu amaca yönelik kurgulanan senaryo sayısı 6 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında 2 dakika 51 saniye sonunda süreç tamamlanmıştır. Optimizasyon işlemi sona erdikten sonra çalıştırılan otomatik testlerden 4 tanesi başarılı, diğer 2 tanesi de başarısız olarak sonuçlanmıştır. Bu modül işletilmeden önce otomatize olmayan VTS021, VTS025 senaryolarının testleri manuel olarak gerçekleştirilmelidir. Bu durum otomatik testlerin işletilmesinde manuel testin gerçekleştirilmesini zorunlu kılmaktadır.

Test Results	
VakıfbankKrediTaksitleriModulu (6)	
✗ VakıfbankKrediAlanınaDegerGir	26 sec
✗ VakıfbankKrediHesaplama	26 sec
✓ VakıfbankKrediTaksitleriAlanınaTıkla	33 sec
✓ VakıfbankKrediTaksitleriHesaplaButonunaTıkla	28 sec
✓ VakıfbankKrediTaksitleriTutarAlanınaDegerGir	28 sec
✓ VakıfbankKrediTaksitleriVadeAlanınaDegerGir	30 sec

Vakıfbank Nakite Taksit Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Nakite Taksit Hesaplama modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş ve 2 dakika 48 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
VakıfbankNakiteTaksitHesaplamaModulu (5)	
✓ VakıfbankNakiteTaksitHesaplaButounaTıkla	33 sec
✓ VakıfbankNakiteTaksitHesaplama	36 sec
✓ VakıfbankNakiteTaksitHesaplamaAlanınaTıkla	35 sec
✓ VakıfbankNakiteTaksitTutarAlanınaDegerGir	32 sec
✓ VakıfbankNakiteTaksitVadeAlanınaDegerGir	32 sec

Vakıfbank Tahvil Bono Hesaplama Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. Tahvil Bono Hesaplama modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında 2 dakika 46 saniye sonunda süreç tamamlanmıştır. Optimizasyon işlemi sona erdikten sonra çalıştırılan otomatik testlerden 2 tanesi başarılı, diğer 3 tanesi de başarısız olarak sonuçlanmıştır. Bu modül işletilmeden önce otomatize olmayan VTS032, VTS033, VTS035 senaryolarının testleri manuel olarak gerçekleştirilmelidir. Bu durum otomatik testlerin işletilmesinde manuel testin gerçekleştirilmesini zorunlu kılmaktadır.

Test Results	
VakıfbankTahvilBonoHesaplamaModulu (5)	
✗ VakıfbankTahvilBonoHesaplama	32 sec
✓ VakıfbankTahvilBonoHesaplaButonunaTıkla	35 sec
✓ VakıfbankTahvilBonoHesaplamaAlanınaTıkla	32 sec
✗ VakıfbankTahvilBonoIslemTipiAlanınaDegerGir	35 sec
✗ VakıfbankTahvilBonoUrunAdiAlanınaDegerGir	32 sec

c. TCMB Uygulamasının Testlerinin Optimizasyonu

TCMB uygulamasının “TCMB ile Halkbank Döviz Karşılaştırma Modülü” modülü için 5 adet otomatik test senaryosunun optimizasyon çalışması yapılmıştır. Her modül için yapılan optimizasyon işlemi sırasıyla aşağıda belirtilmiştir.

TCMB ile Halkbank Döviz Karşılaştırma Modülü

Senaryolar otomatize edilmiş ve her biri kendi ortamında çalıştırılmıştır. Sonrasında ise tüm senaryolar eşzamanlı yürütülebilir hale getirilmiştir. TCMB ile Halkbank Döviz Karşılaştırma modülünde bu amaca yönelik kurgulanan senaryo sayısı 5 tanedir. Bu senaryolar ardı ardına çalışacak şekilde programlandığında işlem sorunsuz gerçekleşmiş

ve 2 dakika 53 saniye sonunda süreç tamamlanmıştır. Sonuç olarak optimizasyon adımları sonrası her bir test senaryosunun otomatize olarak başarılı bir şekilde işlediği gözlemlenmiştir.

Test Results	
TCMBHalkbankDovizKarsilastirmaModulu (5)	
✓ TCMBHalkbankKurKarsilastirma	1 min
✓ HalkbankEUROKurBilgisi	29 sec
✓ HalkbankUSDKurBilgisi	28 sec
✓ TCMBEUROKurBilgisi	28 sec
✓ TCMBUSDKurBilgisi	28 sec



3.1 Sonuç ve Öneriler

3.1.1 Manuel Yazılım Test Sonuçları

Manuel yazılım test sürecinde de belirtildiği üzere Halkbank, Vakıfbank ve TCMB uygulamalarına ait manuel testler belirlenen fonksiyonel özelliklere göre tamamlanmıştır. Her uygulamaya ait modül bazlı manuel test durumları sırasıyla aşağıda belirtilmiştir.

a. Halkbank Uygulamasının Manuel Testleri

Halkbank uygulamasının manuel testleri, test senaryo oluşturma süreci bölümünde belirlenen test senaryoları baz alınarak gerçekleştirilmiştir. Manuel testler için öncelikle İnternet Şubesi modülündeki test senaryoları kullanılmıştır. İnternet Şubesi modülünde yer alan 10 tane senaryonun testleri gerçekleştirilmiştir. Ardından Kredi Hesaplama modülünde yer alan 4 tane senaryonun testleri gerçekleştirilmiştir. Hesaplama Aracı modülünde yer alan bir tane, Kredi Mevzuat modülündeki altı tane, Mevduat Getirisi modülündeki altı tane, Döviz Çevirici modülündeki altı tane, Fon Tutarı modülündeki beş tane, Hazine Bonosu modülündeki beş tane, Vergi Takvimi modülüne ait bir tane ve Repo Getirisi modülündeki beş tane test senaryosunun testleri gerçekleştirilmiştir.

Tablo 1.1 Halkbank Uygulamasına ait modülüne ait manuel test durumları

	Senaryo Adedi	Test Durumu
İnternet Şubesi	10	Başarılı
Kredi Hesaplama	8	Başarılı
Hesaplama Aracı	1	Başarılı
Kredili Mevzuat	6	Başarılı

Mevduat Getirisi	6	Başarılı
Döviz Çevirici	6	Başarılı
Fon Tutarı	5	Başarılı
Hazine Bonosu	5	Başarılı
Repo Getirisi	5	Başarılı
Vergi Takvimi	1	Başarılı
Toplam Senaryo	53	

Halkbank.com.tr uygulamasına ait 53 test senaryosunun manuel testleri tamamlanmıştır. Tablo 1.1 de test senaryolarına bağlı veriler gösterilmektedir.

b. Vakıfbank Uygulamasının Manuel Testleri

Vakıfbank.com.tr uygulamasının manuel testleri, test senaryo oluşturma süreci bölümünde belirlenen test senaryoları baz alınarak gerçekleştirilmiştir. Manuel testler için öncelikle İnternet Şubesi modülündeki test senaryoları kullanılmıştır. İnternet Şubesi modülünde yer alan 10 tane senaryonun testleri gerçekleştirilmiştir. Ardından Hesaplama Aracı modülünde yer alan bir tane senaryonun testleri gerçekleştirilmiştir. Mevduat Hesaplama modülünde yer alan sekiz tane, Kredi Taksitleri modülünde yer alan altı adet, Nakite Taksit Hesaplama modülünde yer alan beş adet ve Tahvil Bono Hesaplama modülünde yer alan beş adet test senaryosunun testleri de gerçekleştirilmiştir.

Tablo 1.2 Vakıfbank.com.tr modülüne ait manuel test durumları

Modül İsmi	Senaryo Adedi	Test Durumu
İnternet Şubesi	10	Başarılı
Hesaplama Aracı	1	Başarılı

Mevduat Hesaplama	8	Başarılı
Kredi Taksitleri	6	Başarılı
Nakite Taksit Hesaplama	5	Başarılı
Tahvil Bono Hesaplama	5	Başarılı
Toplam Senaryo	35	

Vakıfbank.com.tr uygulamasına ait 35 test senaryosunun manuel testleri gerçekleştirilmiştir. Tablo 1.2 de test senaryolarına bağlı veriler gösterilmektedir.

c. TCMB Uygulamasının Manuel Testleri

TCMB.gov.tr uygulamasının manuel testleri, test senaryo oluşturma süreci bölümünde belirlenen test senaryoları baz alınarak gerçekleştirilmiştir. Bu uygulamada TCMB Halkbank Döviz Karşılaştırma Modülünün test senaryoları kullanılmıştır. Bu modülde yer alan beş tane test senaryosunun testleri gerçekleştirilmiştir.

Modül İsmi	Senaryo Adedi	Test Durumu
TCMB Halkbank Döviz Karşılaştırma Modülü	5	Başarılı
Toplam Senaryo	5	

Tablo 1.3 TCMB.gov.tr module ait manuel test durumları

TCMB.gov.tr uygulamasına ait 5 test senaryosunun manuel testleri gerçekleştirilmiştir. Tablo 1.3 de test senaryolarına bağlı veriler gösterilmektedir.

Halkbank.com.tr, Vakıfbank.com.tr ve TCMB.gov.tr Uygulamaları için test senaryoları baz alınarak ve manuel testler koşularak manuel test süreci gerçekleştirilmiştir. Tablo 1.4 de gerçekleştirilen testler, uygulama adı, test senaryo kodu, senaryo adedi ve test durumları ile ilgili bilgiler gösterilmiştir.

Tablo 1.4 Uygulamaya ait manuel test durumları

Uygulama Adı	Test Senaryo Kodu	Senaryo Adedi	Test Durumu
Halkbank.com.tr	HTS	53	Başarılı
Vakıfbank.com.tr	VTS	35	Başarılı
TCMB.gov.tr	TTS	5	Başarılı
Toplam Senaryo	3	93	

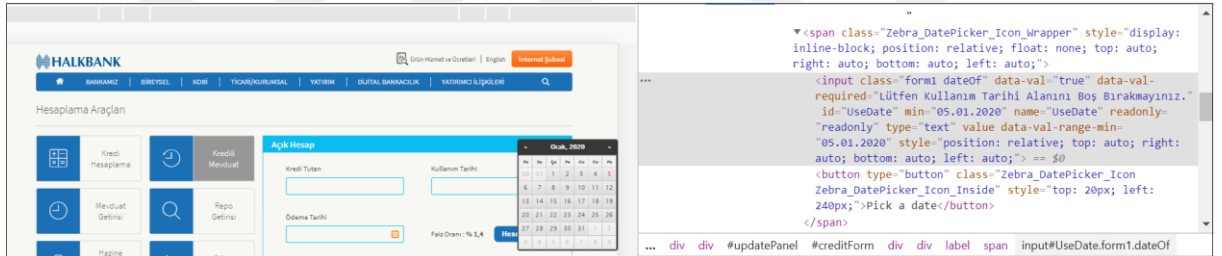
3.1.2 Selenium Web Driver ile Uygulama Adına Göre Otomatize Edilen ve Edilemeyen Test Senaryoları

Halkbank uygulaması için oluşturulan 53 adet test senaryosundan (% 94) 50 sinin otomotizasyonu gerçekleştirildi, 3 tane senaryonun otomotizasyonu ise gerçekleştirilememiştir. Vakıfbank uygulaması için oluşturulan 34 adet test senaryosundan (% 76,5) 27 sinin otomotizasyonu gerçekleştirildi, 8 adet senaryonun otomotizasyonu ise gerçekleştirilememiştir. TCMB uygulaması için oluşturulan 5 tane test senaryosundan **tamamının** otomotizasyonu gerçekleştirilmiştir. Toplamda ise 92 senaryodan (86,5) 81' i otomatize edildi, 11 adet senaryo ise otomatize edilememiştir.

Tablo 1.5 Uygulama Adına Göre Otomatize Edilen ve Edilemeyen Senaryolar

Uygulama Adı	Yazılan Senaryo Sayısı	Otomatize edilen Senaryo Sayısı	Otomatize edilemeyen Senaryo Sayısı
Halkbank	53	50- (%94)	3-(%6)
Vakıfbank	35	27- (%70,4)	8 - (%29,6)
TCMB	5	5 - (%100)	0
Toplam	92	81 - (%86,5)	11 - (%13,5)

Halkbank uygulamasında Kredili Mevduat Modülünde 3 adet senaryo otomatize edilememiştir. HTS022, HTS023, HTS025 numaralı senaryolar otomatize edilemeyen senaryolardır. Örnek olarak HTS022 senaryosu, "Kredili Mevduat" menüsüne tıklandıktan sonra çıkan pencerede "Kullanım Tarihi" alanına tıklanarak veri girilmesinin sağlanmasıydı. Burada kullanım tarihi alanına tıklanmasına rağmen, kodda belirtilen tarih seçimi yapılamamasından kaynaklı problem yaşanmaktadır. HTS023 ile HTS025 senaryolarında da HTS022 senaryosunda olduğu gibi aynı sorunla karşılaşmıştır.

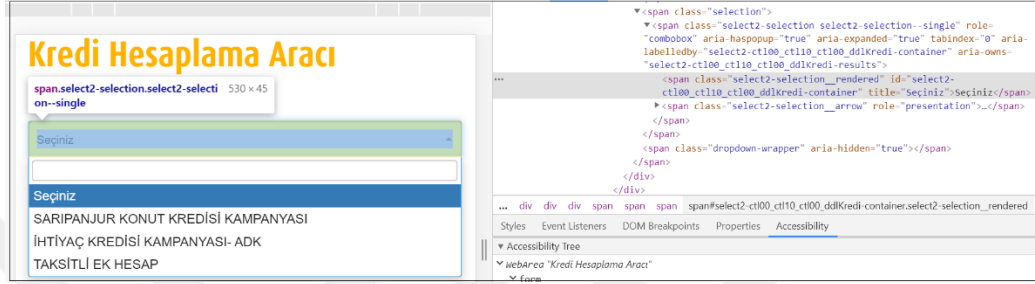


Şekil 3.1 Halkbank Kredili Mevduat Ekranı

```
[TestMethod]
public void HalkbankKullanımTarihiAlanınaDegerGir()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.halkbank.com.tr/");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    driver.FindElement(By.LinkText("Hesaplama Aracı")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.CssSelector("#content > div > div.col-md-5.hesaplamaIcon > a:nth-child(2) > span")).Click();
    Thread.Sleep(1000);
    driver.FindElement(By.Id("UseDate")).Click();
    driver.FindElement(By.Id("UseDate")).SendKeys("30.04.2020");
    Thread.Sleep(1000);
    Console.WriteLine("Kullanım Tarihi Alanına Değer Girildi");
    Thread.Sleep(1000);
    driver.Close();
    driver.Quit();
}
```

Şekil 3.2 HTS022 Numaralı Otomasyon Test Senaryo Kodu

Vakıfbank uygulamasında Kredi Taksitleri Modülünde 2 adet senaryo otomatize edilememiştir. VTS021, VTS025 numaralı senaryolar otomatize edilemeyen senaryolardır. Örnek olarak VTS021 senaryosu, "Kredi Taksitleri" menüsüne tıklandıktan sonra çıkan pencerede " Kredi" combobox'ından veri seçilebilmesinin sağlanmasıydı. Burada Kredi combobox'ına tıklanıyor, ancak kodda da belirtildiği gibi combobox'dan veri seçimi yapamamasından kaynaklı problem yaşanmaktadır. VTS025 numaralı senaryoda da VTS021 senaryosunda olduğu gibi aynı sorunla karşılaşmıştır.



Şekil 3.3 Vakıfbank Kredi Hesaplama Ekranı

```
[TestMethod]
public void VakıfbankKrediAlaninaDegerGir()
{
    ChromeDriver driver = new ChromeDriver();
    driver.Navigate().GoToUrl("https://www.vakifbank.com.tr/kredi-hesaplama-araci.aspx?pageID=1333");
    Thread.Sleep(1000);
    driver.Manage().Window.Maximize();
    SelectElement secim = new SelectElement(driver.FindElement(By.CssSelector("#select2-ctl00_ctl100_ctl100_ddlKredi-container")));
    //secim.SelectByText("TAKSİTLİ EK HESAP");
    secim.SelectByIndex(3);
    // driver.FindElement(By.CssSelector("#select2-ctl00_ctl100_ctl100_ddlKredi-container")).Click();
    // driver.FindElement(By.CssSelector("#select2-ctl00_ctl100_ctl100_ddlKredi-container")).SendKeys("TAKSİTLİ EK HESAP");
    //driver.FindElement(By.Id("select2-ctl00_ctl100_ctl100_ddlKredi-container")).SendKeys("TAKSİTLİ EK HESAP");
    Console.WriteLine("Kredi Taksitleri Kredi Alanina Deger Girildi");
    driver.Close();
    driver.Quit();
}
```

Şekil 3.4 VTS021 Numaralı Otomasyon Test Senaryo Kodu

Vakıfbank uygulamasında Mevduat Hesaplama Modülünde 4 adet senaryo otomatize edilememiştir. VTS015, VTS017, VTS018, VTS019 numaralı senaryolar otomatize edilemeyen senaryolardır. VTS015, VTS018 ve VTS019 numaralı senaryolar da VTS021 senaryosunda olduğu gibi aynı sorunla karşılaşmıştır.

Vakıfbank uygulamasında Tahvil Bono Hesaplama Modülünde 3 adet senaryo otomatize edilememiştir. VTS032, VTS033, VTS035 numaralı senaryolar otomatize edilemeyen senaryolardır. Bu senaryoların tamamı VTS021 senaryosunda olduğu gibi aynı sorunla karşılaşmıştır.

Ayrıca otomotizasyonu gerçekleştirilemeyen farklı senaryolarla da karşılaşmıştır. Test senaryolarının otomotizasyonu gerçekleştirilmesi esnasında ilgili bölümler seçim, click,

data giriři, date seçimi gibi durumlar için id ve class kullanılıp, senaryolar gerçekleştirilmeye çalışılmıştır. Id ve class kullandığında otomatize edilemeyen senaryolar için CssSelector ve Xpath kullanıldığında senaryoların başarılı şekilde çalıştığı gözlemlenmiştir.

3.1.3 Selenium Web Driver Aracı Test Otomasyon Sonuçları

Halkbank uygulaması için yazılan 53 adet senaryo çalıştırıldığında 24 dakika 45 saniye gibi bir zamanda otomatize edilen test senaryolarının testleri gerçekleştirilmiştir. Vakıfbank uygulaması için yazılan 34 adet senaryo çalıştırıldığında 17 dakika 10 saniye gibi bir zamanda otomasyon senaryolarının testleri gerçekleştirilmiştir. TCMB uygulaması için yazılan 5 adet senaryo çalıştırıldığında 2 dakika 53 saniye gibi bir zamanda otomasyon senaryolarının testleri gerçekleştirilmiştir. Toplamda ise 92 adet senaryo çalıştırıldığında 44 dakika 48 saniye gibi bir zamanda otomasyon senaryolarının testleri gerçekleştirilmiştir.

Tablo 1.6 Selenium Web Driver Test Otomasyon Sonuçları

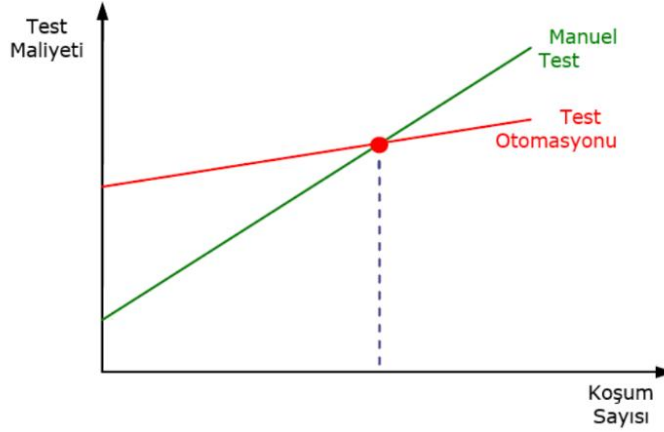
Uygulama Adı	Modül Adı	Senaryo Sayısı	Süre(dk:sn)
Halkbank	İnternet Şubesi	10	5:47
Halkbank	Hesaplama Aracı	1	0:31
Halkbank	Kredi Hesaplama	8	3:53
Halkbank	Kredili Mevduat	6	3:21
Halkbank	Mevduat Getirisi	6	3:10
Halkbank	Repo Getirisi	5	0:46
Halkbank	Hazine Bonosu	5	0:49
Halkbank	Döviz Çevirici	6	3:15
Halkbank	Vergi Takvimi	1	0:29
Halkbank	Fon Tutarı	5	2:44
HTS - Toplam	10	53	24:45
Vakıfbank	İnternet Bankacılığı	10	4:39
Vakıfbank	Hesaplama Aracı	1	0:33
Vakıfbank	Mevduat Hesaplama	8	3:33
Vakıfbank	Kredi Taksitleri	6	2:51
Vakıfbank	Nakite Taksit Hesaplama	5	2:48
Vakıfbank	Tahvil Bono	5	2:46
VTS - Toplam	6	35	17:10
TCMB	TCMB ile Halkbank Döviz Karşılaştırma	5	2:53
TTS - Toplam	1	5	2:53
Genel Toplam	17	92	44:48

3.1.4 Test Otomasyonu ve Manuel Testin Karşılaştırılması

Manuel test, Manuel test, bir tester'in son kullanıcı rolünü oynadığı ve uygulamanın işlevselliğinin kullanıcı gereksinimlerine uygun olup olmadığını test ettiği bir yaklaşımdır. Tester'in test stratejisini planlaması gerekir. Test caseleri manuel yazılmalıdır ve sonra test dataları hazırlanmalıdır. Bu test caselerinin yürütülmesi ve aynı zamanda elde edilen sonuçların beklenen sonuçlar ile eşleşen olup olmadığını kontrol de tester tarafından manuel yapılmalıdır. Bu nedenle, manuel test çok zahmetli ve zaman alıcı bir görevdir. Tüm görevler yalnızca tester tarafından yapılması gerektiği için hatalar ortaya çıkabilir. Tester'in uygulamanın işlevselliği hakkında bilgili ve test caselerini hazırlamak için çok yetenekli olması gerekir. Test otomasyonu ise önceden tahmin edilmiş sonuçlarla gerçek sonuçların karşılaştırılması ve testlerin koşulmasını kontrol etmek için (test edilmiş yazılımdan farklı olan) belirli yazılımın kullanılmasıdır. Otomasyon testinde de test caseleri öncelikle manuel script olarak yazılır, ancak bu scriptlerin yürütülmesi kullanılan tool aracılığıyla otomatikleştirilir. Otomasyon testi test caselerinin birçok kez tekrar edilmesi gerektiğinde çok yararlıdır. Sistem regresyon testleri sırasında, testlerin daha sık koşturulabilmesi gerektiğinde otomatik testler manuel testten daha doğru sonuçlar doğurur. Regresyon testinde, uygulamada bir değişiklik yapıldığında, bir uygulamada aynı test caseleri tekrar tekrar yürütülür. Bu gibi durumlarda, scriptlerin yürütülmesini manuel gerçekleştirmek çok zaman alabilir ve burada hata yapma riski de oldukça yükselir. Bu nedenle, organizasyon gereksinimleri sık sık değiştiğinde veya bazı script setinin tekrar tekrar uygulanması gerektiğinde otomasyon testinin daha hızlı ve güvenilir olduğu görülür. Böylelikle insan kaynaklarına yapılan yatırım, test otomasyonu ile azalmaktadır. Test aracı ve lisans maliyeti olmaması açısından ise manuel test avantajlıdır.

Çalışmamızda bir test kaynağı ile tüm test senaryolarının manuel olarak gerçekleşme süresinin ortalama değeri 1 adam gün olduğu gözlemlenmiştir. Bu sürenin saat karşılığı (1*8) 8 saate yakın bir zamandır. Otomasyon testi için bir test kaynağı ile test senaryolarının otomatize edilme süresinin ortalama 15 adam gün olduğu gözlemlenmiştir. 15 adam gün, yaklaşık olarak (15*8) 120 saate tekabül etmektedir. Ancak bu zaman en başta test scriptlerinin gerçekleştirilmesi için kullanılır. Akabinde testler her iterasyonda otomatik işletildiğinde ortalama 45 dakika zaman almaktadır. Yukarıda da bahsedildiği üzere otomasyon testi, test caselerinin birçok kez tekrar edilmesi gerektiğinde çok yararlıdır. İki haftada bir regresyon testi koşulması gerektiği

durumlar dikkate alındığında manuel testler için her iki haftada 1 gün (8 saat) zaman harcanmaktadır. Ancak testlerin otomatizasyonunun olduğu durumda her iki haftada harcanması gereken zaman ortalama 45 dakika ve otomatik testlerin maintenance'ı ile beraber bu süre maksimum 2 saat olmaktadır. Bu case için yazılım test otomasyonu ile manuel test kıyaslandığında yazılım test otomasyonunun önde olduğu görülmektedir.



Şekil 3.5 Test Maliyeti Koşum Sayısı Grafiği

3.1.5 Selenium IDE ile Selenium Web Driver'ın Karşılaştırılması

Selenium paketinin en basit çerçevesidir ve öğrenmesi en kolay olandır. Ancak, sadeliği nedeniyle, Selenium IDE sadece bir prototip aracı olarak kullanılır ve daha sonra Web Driver'a aktarabileceğiniz basit test örnekleri ve test paketleri oluşturmak için kullanılır.

Selenium IDE “Kayıt/Oynatma” özelliği sayesinde kullanıcının web sayfasında gerçekleştirilen tüm kullanıcı eylemlerini kaydederek test senaryoların otomatikleştirmesini sağlar. Kayıt oturumu tamamlandıktan sonra, tüm eylemleri yeniden oynatarak, web sayfaları sürekli olarak test edilebilir. Test otomasyonu komut dosyaları yazmak için kullanıcıların programlama hakkında herhangi bir teknik bilgiye sahip olması gerekmez. Sadeliği nedeniyle, Kayıt/Tekrar çerçeveleri sınırlamaları ne olursa olsun tüm kullanıcılar için caziptir.

Selenium IDE’de programlama da ön bilgi olmadan ya da çok az bilgiyle testler oluşturulur. Web Driver’da ise programlanma bilgisi gereklidir.

Web Driver ayrıca test scriptlerinizi oluştururken farklı programlama dilleri kullanmanıza olanak tanır. Selenium IDE’de mümkün değildir.

Web Driver farklı tarayıcılara karşı test yürütmek için izin veren bir web otomasyon çerçevesi, Selenium IDE ise sadece Firefox, Chrome tarayıcılarını desteklemektedir.

Daha gelişmiş test caseleri oluşturmak için Selenium Web Driver kullanılır. Selenium IDE sadece bir prototip aracı olarak kullanılmakta olup, gelişmiş test caselerini karşılayamamaktadır. TTS kodu ile başlayan senaryolar Selenium IDE ile gerçekleştirilememiştir.

Selenium Web Driver, özelleştirilmiş test sonuçları oluşturmak için kullanışlıdır. Selenium IDE ile bunu karşılamak oldukça zordur. TTS kodu ile başlayan senaryolar özelleştirilmiş olduğundan Web Driver ile gerçekleştirilmiştir.

Selenium IDE’de Kayıt/Oynatma çerçevelerinin, testin yürütülmesinde tutarlılık problemi ve kodun yeniden kullanılabilirlik eksiklikleri nedeniyle bir dizi sınırlaması vardır. Asynchronicity dinamik web uygulamalarında önemli bir özelliktir. Sunucuya eşzamanlı çağrılar yapılması, sayfadaki her öğenin durumunu değiştirebilir. Selenium IDE’de kayıt/yeniden oynatma çerçevelerinin dinamik durum değişiminde sorunlar olduğu görülmektedir. Örneğin, gerçek senaryoda komut dosyasının tetikleyici olması için iletişim kutusunun kaybolmasını beklemesi gerekir, ancak sayfada bekleme iletişim kutusunu algılayamaz ve sayfada yanlış öğeye ulaştığından, bu nedenle test çalışması başarısız olur. Yani senkronizasyon sorunları vardır.

Selenium IDE’de kodun yeniden kullanılabilirliği, kayıt/oynatma test çerçevelerinin bir diğer dezavantajıdır. Kullanıcı eylemlerini kaydederken, kayıt/yeniden oynatma çerçeveleri çerçeveye özgü kod segmentleri oluşturur. Bu nedenle, test çerçevesi değiştirildiğinde oluşturulan komut dosyası kodları güncelleştirilmelidir.

Sonuç olarak otomasyon, “Kayıt ve Oynatma” eylemlerinden daha fazlasıdır. Yazılım yazma hakkında bildiğimiz tüm kurallar test otomasyonu için de geçerlidir. Test scriptleri, “Kendinizi Tekrar Etmeyin” gibi en azından basit kodlama ilkeleri uygulayarak tasarlanmalıdır. Kaynak kodları her sürümde değiştiğinden, test scriptleri buna göre güncelleştirilmelidir. Bakım maliyetini azaltabilmek amacıyla test otomasyonu scriptlerini güncellememiz gerektiğinde, yalnızca bir modülü, sınıfı veya makroyu, belki de yalnızca değişken bir değeri değiştirmemiz gerekir. Bu bakış açısı maliyeti önemli ölçüde azaltmaktadır. Örneğin, bir nesne kütüphanesi ve fonksiyon kütüphanesi oluştururuz ve yeniden kullanılabilir test nesneleri ve fonksiyonlarını oraya

koyarız. Bir test scriptine bakım yaparken, mümkün olduğunca ortak kütüphanelerde değişiklik yapmaya çalışılır. Ortak test kütüphaneleri üretim kodu ve gereksinimleriyle eşleştikçe yapılandırmaya çalışılır. Test senaryosunu yeniden kaydetmek gerekmez. Web Driver ile bu kısım gerçekleştirilir. Ama Selenium IDE ile mümkün değildir.

3.2 Sonuçlar

Çalışmada manuel ve otomatik test süreçleri karşılaştırıldığında test otomasyonu ile otomatizasyonu gerçekleştirilen testlerin ileriki süreçlerde zamandan ciddi oranda kazanç sağladığı sonucuna varılmıştır. Test süreçlerinde ileriki zamanlarda daha az insan kaynağına gereksinim duyulduğu görülmektedir. Test otomasyon ve manuel test karşılaştırılmıştır. Selenium IDE ve Selenium Web Driver araçları kullanılarak test otomasyon çalışması yapılmıştır ve arasındaki farklar kıyaslanmıştır. Selenium Web Driver kullanılarak otomasyon test senaryoların başarı yüzdesi hesaplanmıştır.

- [1] Yazgelistir, Eriřim Tarihi: 4 Eylöl 2019, <http://www.yazgelistir.com/makale/proje-yonetimsurecleri>
- [2] Salyangoz, Eriřim Tarihi: 7 Ekim 2019, <http://salyangoz.com.tr/blog/2013/11/02/digerleri/yazilim-gelistirme-surec-modelleri-nelerdir/>
- [3] Innova, Eriřim Tarihi: 8 Ekim 2019, <http://www.innova.com.tr/uygulama-gelistirme-yasamdongusu.asp>
- [4] Harshada Kekare, buzzle.com: <http://www.buzzle.com>, 2019
- [5] Medium, Eriřim Tarihi: 13 Haziran 2019, <https://medium.com/@ozaneseriu/yazilim-test-sevi%CC%87yeleri%CC%87-593a17a16d7f>
- [6] Wikipedia, Eriřim Tarihi: 1 Ocak 2020, https://tr.wikipedia.org/wiki/Regresyon_testi
- [7] Khan, M. E., and Khan, F., "A comparative study of white box, black box and grey box testing techniques," Int. J. Adv. Comput. Sci. Appl, 3(6), 2012
- [8] MSDN, Eriřim Tarihi: 13 Haziran 2019, [http://msdn.microsoft.com/enus/library/ee417692\(v=vs.85\).aspx](http://msdn.microsoft.com/enus/library/ee417692(v=vs.85).aspx)
- [9] Devnot, Eriřim Tarihi: 15 Temmuz 2019, <http://devnot.com/2015/test-otomasyon-araci-seleniumu-taniyalim/>
- [10]Guru99, Eriřim Tarihi: 15 Ağustos 2019, <https://www.guru99.com/>

TEZDEN ÜRETİLMİŞ YAYINLAR

İletişim Bilgisi: info@orporganizasyon.com

Konferans Bildirileri

1. 5. Uluslararası Mühendislik Mimarlık ve Tasarım Kongresi (Yazılım Yaşam Döngüsünde Testin Önemi ve Selenium Kullanılarak Bir Test Otomasyonunun Gerçekleştirilmesi)

