Test Otomasyon Uygulaması

Yazılım Mühendisliği Ana Bilim Dalı

Yüksek Lisans Bitirme Projesi

Ahmet Ali Can

Bitirme Projesi Danışmanı: Dr. Öğr. Üyesi Serpil Yılmaz

Haziran 2023

İzmir Kâtip Çelebi Üniversitesi Fen Bilimleri Enstitüsü öğrencisi **Ahmet Ali Can** tarafından hazırlanan **Test Otomasyon Uygulaması** başlıklı bu çalışma tarafımızca okunmuş olup, kapsam ve nitelik açısından başarılı bulunarak tarafımdan YÜKSEK LİSANS BİTİRME PROJESİ olarak kabul edilmiştir.

ONAYLAYANLAR:

Bitirme Projesi Danışmanı: **Dr. Öğr. Üyesi Serpil Yılmaz**
İzmir Kâtip Çelebi Üniversitesi

Yazarlık Beyanı

Ben, **Ahmet Ali Can**, başlığı **Test Otomasyon Uygulaması** olan bu bitirme projesinin içinde sunulan bilgilerin şahsıma ait olduğunu beyan ederim. Ayrıca:

- Bu çalışmanın bütünü veya esası bu üniversitede Yüksek Lisans derecesi elde etmek üzere çalıştığım süre içinde gerçekleştirilmiştir.
- Daha önce bu bitirme projesinin herhangi bir kısmı başka bir derece veya yeterlik almak üzere bu üniversiteye veya başka bir kuruma sunulduysa bu açık biçimde ifade edilmiştir.
- Başkalarının yayımlanmış çalışmalarına başvurduğum durumlarda bu çalışmalara açık biçimde atıfta bulundum.
- Başkalarının çalışmalarından alıntıladığımda kaynağı her zaman belirttim. Bitirme projesinin bu alıntılar dışında kalan kısmı tümüyle benim kendi çalışmamdır.
- Kayda değer yardım aldığım bütün kaynaklara teşekkür ettim.
- Bitirme projesinde başkalarıyla birlikte gerçekleştirilen çalışmalar varsa onların katkısını ve kendi yaptıklarımı tam olarak açıkladım.

Tarih: 05.05.2023

Test Otomasyon Uygulaması

Öz

Bu çalışmada yazılım testi içerisinde gerçekleştirilen sürekli testler için otomasyon kütüphanesi geliştirilmiştir. Otomasyon dünyasında yer alan uygulamaların haricinde kendi içinde kod yazımı olmadan kullanımı ve geliştirmek isteyen kullanıcılar için de anlaşılır ve sade bir yapıda geliştirilmiş bir kütüphanedir. Selenium IDE'sinde yer alan fonksiyonlar analiz edilip, kullanıcının kod yazma gerekliliğini kaldırarak sade ve geliştirilebilir metotlarla otomasyonlar hazırlanabilecektir. Ayrıca açık kaynaklı olduğundan dolayı geliştiriciler tarafından metotlar düzenlenip farklı projelerde de kullanılabilir.

Anahtar Sözcükler: Mühendislik, teknoloji, test otomasyonu, manuel test, yazılım

Test Automation Application

Abstract

In this study, an automation library has been developed for continuous tests carried out in software testing. Apart from the applications in the automation world, it is a library developed in a clear and simple structure for users who want to use and develop it without writing code. By analyzing the functions in the Selenium IDE, automations can be prepared with simple and developable methods by removing the user's need to write code. In addition, since it is open source, the methods can be edited by the developers and used in different projects.

Keywords: Engineering, technology, test automation, manual testing, software

Teşekkür

Saygıdeğer Dr.Öğr.Üyesi Serpil Yılmaz ,

Bitirme projemi tamamladığım bu önemli aşamada, sizinle paylaşmak için size bu teşekkür yazısını iletmek istiyorum. İlgili rehberliğiniz, destekleriniz ve değerli bilgileriniz için size içtenlikle teşekkür etmek istiyorum. Ayrıca, sabır ve özenle proje çalışmamı okuyarak ve her soruma içtenlikle yanıt vererek gösterdiğiniz destek için minnettarım.

Aynı şekilde ihtiyaç duyguğum her an desteklerini esirgemeyen ve bana yardımcı olmak için uğraştığı için Doç. Dr. Aytuğ Onan ‘ a da teşekkür etmek istiyorum.

İçindekiler

Tablolar Listesi

Şekiller Listesi

Kısaltmalar Listesi

VS	Visual Studio
FBE	Fen Bilimleri Enstitüsü
İKÇÜ	İzmir Kâtip Çelebi Üniversitesi
ORCID	Open Researcher and Contributor ID
IDE	Integrated Development Environment
QA	Quality Assurance
OOP	Object-Oriented Programming
C#	C Sharp

Bölüm 1

1.1 Giriş

Bu bölümde projede kullanılan program ve IDE ler hakkında genel bilgilendirme yapılmıştır.

1.2 Visual Studio

Visual Studio, Microsoft tarafından geliştirilen ve popüler bir tümleşik geliştirme ortamıdır (IDE). Yazılım geliştiricilerin çeşitli programlama dilleriyle uygulama oluşturmalarını, test etmelerini, hata ayıklamalarını ve dağıtmalarını sağlar.

Visual Studio, geniş bir programlama dilini destekler ve bu dillerle çalışmak için özelleştirilmiş araçlar sunar. C#, C++, Visual Basic, F#, Python, JavaScript, TypeScript gibi birçok dil ile uyumludur.

Bu geliştirme ortamı, zengin bir özellik seti sunar. Kod editörü, hata ayıklama araçları, sürükle ve bırak form tasarımı, kod tamamlama, derleme, bağlama, test etme, takım işbirliği özellikleri ve daha fazlası gibi birçok araç ve özelliği içerir.

Visual Studio, Windows işletim sistemi için geliştirme yapmak için kullanılan bir araçtır, ancak çeşitli platformlara (web, mobil, bulut, oyunlar vb.) yönelik uygulama geliştirme imkanı sağlar. Örneğin, Xamarin ile mobil uygulamalar, ASP.NET ile web uygulamaları, Unity ile oyunlar gibi farklı alanlarda projeler oluşturulabilir.

Ek olarak, Visual Studio ekosistemi geniş bir şekilde desteklenir ve geliştiricilere geniş bir eklenti ve uzantı koleksiyonu sunar. Bu, geliştiricilerin kendi ihtiyaçlarına göre özelleştirilmiş bir geliştirme deneyimi yaşamalarını sağlar.

Sonuç olarak, Visual Studio, yazılım geliştirme sürecini destekleyen güçlü bir tümleşik geliştirme ortamıdır. Geniş dil desteği, zengin özellik seti ve genişletilebilirlik ile geliştiricilere verimli bir çalışma ortamı sunar [1].

1.3 C#

C#, Microsoft tarafından geliştirilen, nesne yönelimli bir programlama dilidir. C# (C Sharp), .NET platformuna özgüdür ve genellikle Windows uygulamaları, web uygulamaları, oyunlar ve diğer çeşitli yazılımların geliştirilmesinde kullanılır.

C#, C ve C++ gibi dillerden etkilenmiştir, ancak daha modern ve daha kullanıcı dostu bir sözdizimine sahiptir. C#'ın nesne yönelimli programlama (OOP) prensiplerine tam destek verdiği için, kodun düzenlenmesini, bakımını ve genişletilmesini kolaylaştırır.

C# dilinin bazı özellikleri şunlardır:

1. Nesne Yönelimli Programlama (OOP): C#, nesne yönelimli programlamanın temel prensiplerini destekler. Sınıflar, nesneler ve kalıtım gibi OOP kavramları, kodun düzenlenmesini ve yeniden kullanılmasını kolaylaştırır.

2. Tip Güvenliği: C#, derleme zamanında tür güvenliği sağlar. Bu, hataların daha erken tespit edilmesini ve çalışma zamanındaki hataların azalmasını sağlar.

3. Geniş Kütüphane Desteği: C#, .NET Framework veya .NET Core gibi geniş bir kütüphane koleksiyonuna erişim sağlar. Bu kütüphaneler, çeşitli işlevleri yerine getirmek için kullanılabilen hazır bileşenleri içerir.

4. Platform Bağımsızlık: C#, .NET platformu üzerinde çalışır ve bu da onu platform bağımsız hale getirir. Aynı C# kodu, farklı işletim sistemlerinde (Windows, Linux, macOS) çalışabilir.

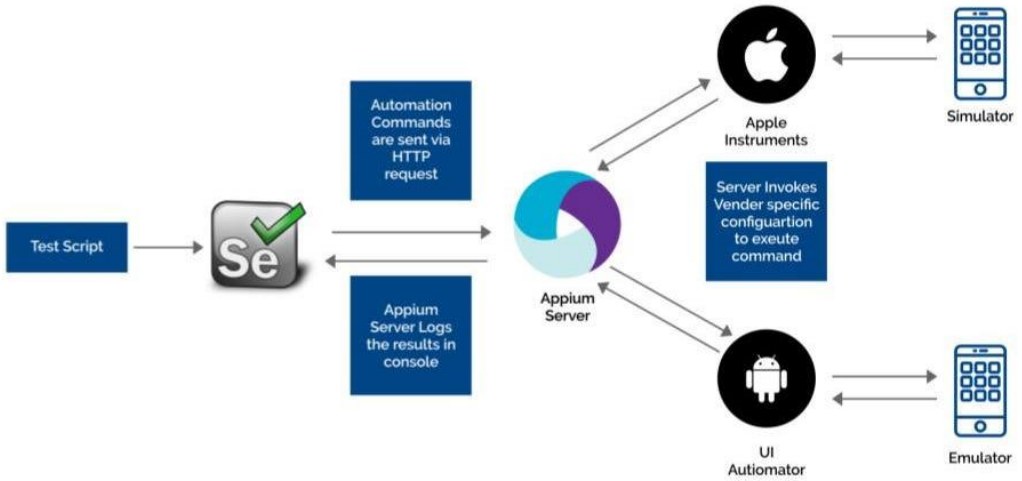
5. Güçlü Hata Ayıklama ve Hata İzleme: Visual Studio gibi entegre geliştirme ortamları, C# geliştiricilerine gelişmiş hata ayıklama ve hata izleme araçları sağlar. Bu, hata ayıklama sürecini kolaylaştırır ve hataların daha hızlı tespit edilmesini sağlar.

C# dilinin bu özellikleri, yazılım geliştirme sürecinde verimliliği artırır ve kaliteli uygulamaların oluşturulmasını sağlar. C#, geniş bir geliştirici topluluğu tarafından desteklenir ve sürekli olarak güncellenir [1].

1.4 Test Otomasyon Araçları

Dünya genelinde kullanılan çeşitli test otomasyon araçları kullanılmaktadır. Bunlardan bazıları Selenium , Appium , TestComplete , JUnit , NUnit , Cucumber , Katalon Studio'dur.

1.4.1 Appium



Şekil 1.1 – Appium Diyagram (Kaynak: https://medium.com/automationmaster/appium-mobile-app-automation-tutorial-2-527d6d78998a)

Appium, açık kaynaklı bir test otomasyon aracıdır ve mobil uygulamaların testlerini gerçekleştirmek için kullanılır. Özellikle Android ve iOS platformlarına odaklanır ve bu platformlarda çalışan uygulamaların testini yapmak için kullanılır.

Appium, yazılım geliştirme dilleriyle (Java, C#, Python, vb.) entegre olabilen bir araçtır. Bu sayede geliştiriciler, tercih ettikleri dilde test senaryolarını oluşturabilir ve uygulamaları otomatik olarak test edebilir. Appium, geliştiricilerin bir dil veya platforma bağlı kalmadan mobil uygulamaları test etmelerini sağlar.

Appium, birkaç farklı bileşenden oluşur:

1. Appium Server: Testleri yürütmek için gereken Appium sunucusudur. Mobil cihazlar ve emülatörlerle iletişim kurar ve komutları uygulamaya yönlendirir.

2. Client Kitaplıkları: Appium, farklı programlama dilleriyle kullanılabilen client kitaplıkları sunar. Bu kitaplıklar, test senaryolarını yazmak ve Appium sunucusuna komut göndermek için kullanılır.

3. Inspector: Uygulamaların UI öğelerini ve özelliklerini incelemek için bir araçtır. Inspector, uygulamaların kaynak kodunu analiz eder ve elementlerin benzersiz tanımlayıcılarını (ID, XPath, vb.) sağlar.

Appium, mobil uygulamaların farklı platformlarda (Android, iOS) test edilmesini sağlar ve aşağıdaki özelliklere sahiptir:

1. Gerçek Cihazlar ve Emülatörlerle Çalışma: Appium, gerçek cihazlarda ve emülatörlerde çalışan mobil uygulamaların testini yapabilir. Bu, farklı ortamlarda uygulamaların test edilebilmesini sağlar.

2. Native, Hybrid ve Web Uygulamalarını Destekleme: Appium, native (doğal), hybrid (karışık) ve web tabanlı uygulamaların testlerini gerçekleştirebilir. Bu, farklı uygulama türlerinin testini tek bir araçla yapabilmenizi sağlar.

3. Birden Fazla Dil ve Framework Desteği: Appium, farklı programlama dilleri ve test framework'leriyle entegre çalışabilir. Bu, geliştiricilerin tercih ettikleri dili ve framework'ü kullanarak test senaryolarını yazmalarını sağlar.

4. Paralel Test Yürütme: Appium, aynı anda birden fazla testi paralel olarak yürütebilme yeteneğine sahiptir. Bu, test sürecinin hızını artırır ve zaman tasarrufu sağlar [2].

1.4.2 TestComplete

TestComplete, kapsamlı bir test otomasyon aracıdır ve farklı platformlarda (web, mobil, masaüstü) çalışan uygulamaların testlerini gerçekleştirmek için kullanılır. Geliştiricilere, test mühendislerine ve QA ekiplerine geniş kapsamlı test otomasyonu imkanı sunar.

TestComplete, aşağıdaki özelliklere sahip bir araçtır:

1. Çoklu Platform Desteği: TestComplete, web, mobil ve masaüstü uygulamalarının testlerini otomatikleştirebilir. Bu, farklı platformlarda çalışan uygulamaların testlerini tek bir araçla yapabilmenizi sağlar.

2. Kolay Kullanıcı Arabirimi: TestComplete, sürükle-bırak özelliğiyle kolay bir kullanıcı arabirimine sahiptir. Test senaryolarını hızlı bir şekilde oluşturmanızı ve testlerinizi sürdürmenizi sağlar.

3. Çeşitli Programlama Dili Desteği: TestComplete, farklı programlama dilleriyle (C#, VBScript, JavaScript, Python, vb.) entegre olabilir. Bu sayede, tercih ettiğiniz dili kullanarak test senaryolarınızı oluşturabilir ve uygulamaları otomatik olarak test edebilirsiniz.

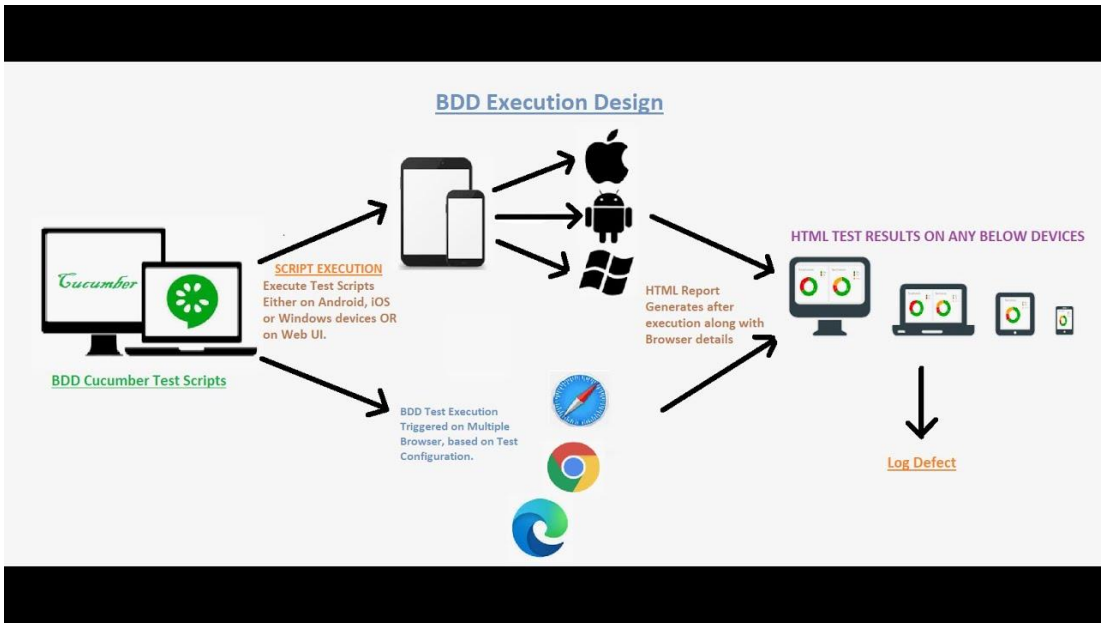
4. Hata Ayıklama ve Kayıt: TestComplete, hata ayıklama işlevleriyle donatılmıştır. Test senaryolarını adım adım izleyebilir, hataları tespit edebilir ve sorunları çözebilirsiniz. Ayrıca, testleri kaydedebilir ve daha sonra yeniden oynatabilirsiniz.

5. Test Raporlama ve Analiz: TestComplete, kapsamlı raporlama ve analiz özelliklerine sahiptir. Test sonuçlarını ayrıntılı bir şekilde görüntüleyebilir, test metriklerini analiz edebilir ve kalite iyileştirmelerine odaklanabilirsiniz.

6. Entegrasyon Yetenekleri: TestComplete, diğer geliştirme araçlarıyla (CI,CD araçları, sürüm kontrol sistemleri, hata takibi araçları) entegre olabilir. Bu, otomatik test sürecini diğer geliştirme süreçleriyle bütünleştirmenizi sağlar.

TestComplete, geniş bir test otomasyon aracı olması ve çeşitli platformlarda kullanılabilmesiyle tanınır. Çoklu platform desteği ve kolay kullanıcı arabirimi sayesinde kullanıcılar, farklı türdeki uygulamaları otomatik olarak test etmek için hızlı ve etkili bir şekilde senaryolar oluşturabilir ve yönetebilir.

1.4.3 Cucumber



Şekil 1.2 – Cucumber Diyagram (Kaynak: https://www.youtube.com/watch?v=X0MdVNiina4)

Cucumber, Behavior-Driven Development (BDD) yaklaşımını destekleyen bir test otomasyon aracıdır. BDD, yazılım geliştirme sürecinde iş gereksinimlerini anlaşılır bir dilde ifade etmeyi ve geliştirici, testçi ve iş paydaşları arasında daha iyi bir işbirliği sağlamayı amaçlar.

Cucumber, uygulamanın beklenen davranışını tanımlayan senaryoları doğal dilde yazmanıza olanak tanır. Bu senaryolar, önceden belirlenmiş kalıplara (Given, When, Then) göre yazılır ve işlevselliği, kabul kriterlerini ve beklenen sonuçları açıkça ifade eder. Bu sayede, taraflar arasında birlikte çalışılabilir ve anlaşılabilir bir dokümantasyon sağlanır.

Cucumber, aşağıdaki bileşenleri içerir:

1. Feature Dosyaları: Senaryoların ve senaryoların gruplandığı dosyalardır. Feature dosyaları, uygulamanın farklı özelliklerini ve davranışlarını temsil eder.

2. Senaryolar: Uygulamanın davranışını tanımlayan adımlardır. Her senaryo, belirli bir durumda uygulamanın nasıl davranması gerektiğini anlatır.

3. Step Tanımları: Senaryoları uygulamanın gerçek koduyla ilişkilendiren adımlardır. Bu adımlar, uygulamanın belirli bir durumda ne yapması gerektiğini ifade eder.

4. Test Koşucuları: Senaryoların otomatik olarak yürütülmesini sağlar. Test koşucuları, senaryoları Cucumber tarafından okunabilir hale getirir ve ilgili step tanımlarını çalıştırır.

Cucumber, farklı programlama dilleriyle (Java, C#, Ruby, vb.) uyumlu çalışabilir ve çeşitli test framework'leriyle entegre olabilir. Böylece, geliştiriciler, tercih ettikleri dil ve framework kullanarak senaryoları uygulamanın gerçek koduyla ilişkilendirebilir.

Cucumber'ın faydaları şunlardır:

1. Anlaşılır ve İletişime Açık Senaryolar: Cucumber, doğal dilde yazılan senaryolar sayesinde iş gereksinimlerini anlaşılır bir şekilde ifade etmenizi sağlar. Bu, tüm paydaşların anlayabileceği bir dokümantasyon sağlar.

2. İşbirliği ve İletişim: Cucumber, geliştiriciler, testçiler ve iş paydaşları arasında işbirliğini artırır. Senaryolar, tüm ekip üyeleri arasında bir tartışma ve anlaşma noktası olarak kullanılabilir.

3. Yeniden Kullanılabilir

lik: Senaryolar, tekrar kullanılabilir adımların tanımlanması sayesinde tekrar kullanılabilir olabilir. Bu, senaryoların daha modüler ve sürdürülebilir olmasını sağlar.

4. Otomatik Test Yürütme: Cucumber, senaryoların otomatik olarak yürütülmesini sağlar. Bu, tekrarlayan test süreçlerini otomatikleştirir ve hızlı geri bildirim almanızı sağlar.

Cucumber, BDD yaklaşımını kullanarak iş gereksinimlerini daha anlaşılır bir şekilde ifade etmenizi ve yazılım geliştirme sürecinde işbirliğini artırmanızı sağlayan güçlü bir test otomasyon aracıdır [3].

1.4.4 Katalon Studio

Katalon Studio, kapsamlı bir test otomasyon aracıdır ve web, mobil ve API testleri gibi farklı türdeki uygulamaların testlerini gerçekleştirmek için kullanılır. Katalon Studio'nun özellikleri, kullanıcı dostu arayüzü ve güçlü yetenekleri sayesinde test ekiplerine ve yazılım geliştiricilere kolaylık sağlar.

Katalon Studio'nun özellikleri aşağıdaki gibi özetlenebilir:

1. Çoklu Platform Desteği: Katalon Studio, web, mobil (Android ve iOS) ve API testlerini destekler. Bu, farklı platformlarda çalışan uygulamaların testlerini tek bir araçla gerçekleştirebilmenizi sağlar.

2. Kullanıcı Dostu Arayüz: Katalon Studio'nun kullanıcı dostu arayüzü, kullanıcıların kolayca test senaryolarını oluşturmasını ve yönetmesini sağlar. Sürükle-bırak özelliği ve hazır fonksiyon kütüphaneleri gibi özellikler, test senaryolarını hızlı bir şekilde oluşturmanızı sağlar.

3. Scripting Desteği: Katalon Studio, test senaryolarını Java,Groovy dilinde kodlama yaparak otomatikleştirmenize olanak tanır. Bu, daha karmaşık senaryoları oluşturmanız ve özelleştirilmiş işlevselliği eklemeniz için esneklik sağlar.

4. Nesne Tanımlama ve Kayıt: Katalon Studio, otomatik nesne tanımlama ve kayıt özelliğine sahiptir. Bu özellik, uygulamanın kullanıcı arabirimindeki öğeleri tanımlamanızı ve test senaryolarınızı daha hızlı bir şekilde oluşturmanızı sağlar.

5. Entegrasyon Yetenekleri: Katalon Studio, popüler CI,CD araçları (Jenkins, Azure DevOps, Git, vb.) ve test yönetimi araçları (JIRA, qTest, TestRail, vb.) ile entegre olabilir. Bu, test sürecini diğer geliştirme süreçleriyle entegre etmenizi sağlar.

6. Raporlama ve Analiz: Katalon Studio, detaylı test raporları ve grafiksel analizler sunar. Bu sayede, test sonuçlarını izleyebilir, hataları tespit edebilir ve test kalitesini izleyebilirsiniz.

Katalon Studio, kullanıcı dostu arayüzü ve geniş kapsamlı özellikleriyle test otomasyonunu kolaylaştıran bir araçtır [4].

1.4 NUnit

NUnit, .NET programlama dilleri (C#, Visual Basic, F# vb.) ile kullanılabilmektedir ve .NET Framework ve .NET Core gibi platformlarda desteklenir. NUnit, yazılım projelerinde birim testlerin otomatik olarak yürütülmesini ve sonuçlarının raporlanmasını kolaylaştıran özelliklere sahiptir.

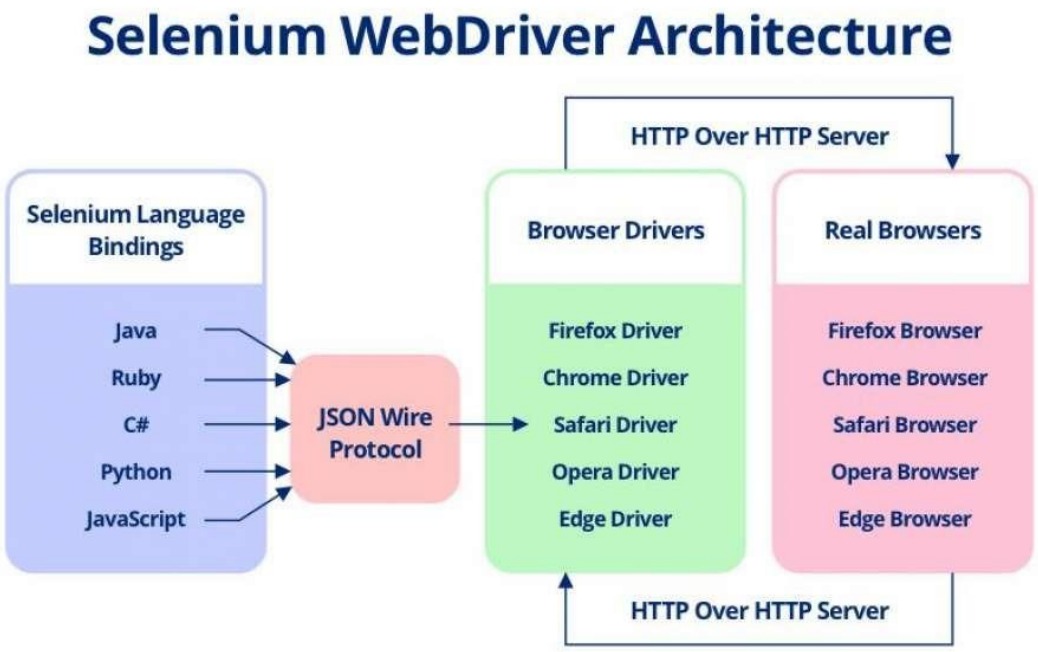
NUnit, yazılım projelerinde test vakalarını tanımlamak için birim test sınıflarını ve test metotlarını kullanır. Bu sınıflar ve metotlar, test edilmek istenen kodun belirli parçalarını hedef alır. NUnit, test vakalarını düzenlemek, testleri gruplandırmak ve çalıştırmak için bir dizi nitelik (attribute) sunar.

NUnit ayrıca, beklenen sonuçları doğrulamak için bir dizi doğrulama (assertion) yöntemi sağlar. Bu yöntemler, testlerin başarılı olup olmadığını belirlemek için kullanılır. Test sonuçları, birim test çerçevesi tarafından sağlanan çıktılar aracılığıyla görselleştirilebilir ve raporlanabilir.

NUnit, yazılım projelerinde test odaklı bir yaklaşımı destekler ve birim testlerin yazılmasını, çalıştırılmasını ve analiz edilmesini kolaylaştırır. Ayrıca, NUnit geniş bir topluluk tarafından desteklenir ve geliştiricilere kullanışlı dokümantasyon ve kaynaklar sağlar.

Sonuç olarak, NUnit, .NET tabanlı yazılım projelerinde birim testlerin oluşturulması ve çalıştırılması için kullanılan popüler bir çerçevedir. Yazılımın doğruluğunu sağlamak ve kalitesini artırmak için birim testlerin kolaylıkla yazılmasını ve yönetilmesini sağlar.

1.5 Selenium IDE



Şekil 1.3 – Selenium Diyagram (Kaynak: https://hackr.io/blog/what-is-selenium-webdriver)

Selenium IDE (Integrated Development Environment), web tabanlı uygulamaların otomasyon testleri için kullanılan bir araçtır. Selenium IDE, kullanıcıların web tarayıcısında kaydedilen adımları oynatmalarını ve tekrarlamalarını sağlayarak otomasyon testlerinin oluşturulmasını kolaylaştırır.

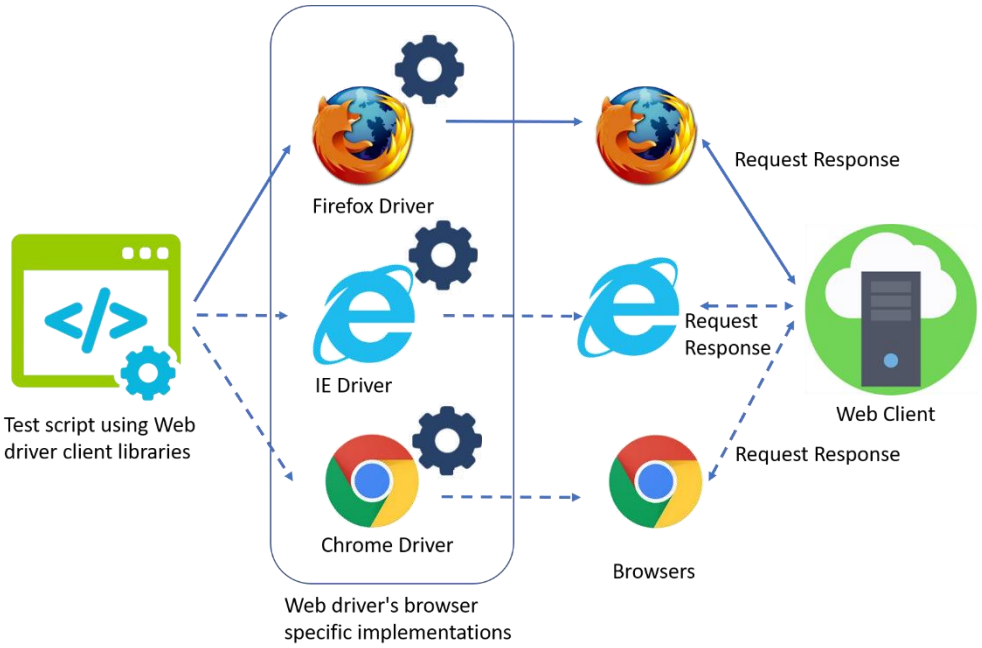
Selenium IDE, Selenium WebDriver'ın bir parçası olarak sunulan bir özelliktir. Bu araç, test senaryolarını kaydetmek ve yönetmek için kullanıcı dostu bir grafik arayüzü sunar. Kaydedilen senaryolar, kullanıcı tarafından düzenlenebilir ve test senaryolarını daha karmaşık hale getirecek ekstra adımlar eklenebilir.

Selenium IDE'nin temel işlevi, web uygulamasında gerçekleştirilen etkileşimleri kaydetmektir. Kullanıcı, tarayıcıda tıklama, metin yazma, form doldurma gibi adımları gerçekleştirerek test senaryosunu kaydedebilir. Kaydedilen senaryo daha sonra otomatik olarak çalıştırılabilir ve tarayıcıda benzer adımların tekrarlanmasını sağlar.

Selenium IDE'nin avantajları arasında kolay kullanılabilir arayüz, hızlı senaryo kaydı, hata ayıklama özelliği ve farklı tarayıcılarla uyumluluk yer alır. Ayrıca, oluşturulan test senaryoları diğer Selenium araçlarıyla entegre edilebilir ve daha geniş kapsamlı test süitleri oluşturmak için kullanılabilir.

Selenium IDE, web tabanlı uygulamaların otomasyon test sürecini hızlandırır ve kolaylaştırır. Kullanıcı dostu arayüzü ve kaydet-çalıştır yaklaşımı sayesinde, geliştiriciler ve test uzmanları, web uygulamalarının doğruluğunu ve işlevselliğini etkin bir şekilde test etmek için Selenium IDE'yi tercih ederler [5].

1.6 Browser Drivers



Şekil 1.4 – Browsers Drivers Diyagram

Browsers drivers, web tarayıcılarının otomasyon testi için kullanılan araçlardır. Bu sürücüler, Selenium WebDriver gibi otomasyon araçlarıyla etkileşim kurarak tarayıcıların kontrolünü sağlar.

Browsers drivers, otomasyon testlerinin tarayıcılarda çalışabilmesi için bir köprü görevi görür. Bu sürücüler, tarayıcının API'lerine erişir ve Selenium WebDriver gibi otomasyon araçlarıyla birlikte kullanılarak tarayıcıyı kontrol etmek için komutlar gönderir [5].

Her tarayıcı için farklı bir sürücü gereklidir. Örneğin, ChromeDriver Google Chrome tarayıcısı için, GeckoDriver Mozilla Firefox tarayıcısı için, WebDriver Safari tarayıcısı için kullanılır. Bu sürücüler, tarayıcının otomatik olarak açılmasını, URL'lerin yüklenmesini, tıklamaların yapılmasını, form doldurmanın gerçekleştirilmesini ve diğer etkileşimleri otomatikleştirmek için kullanılır.

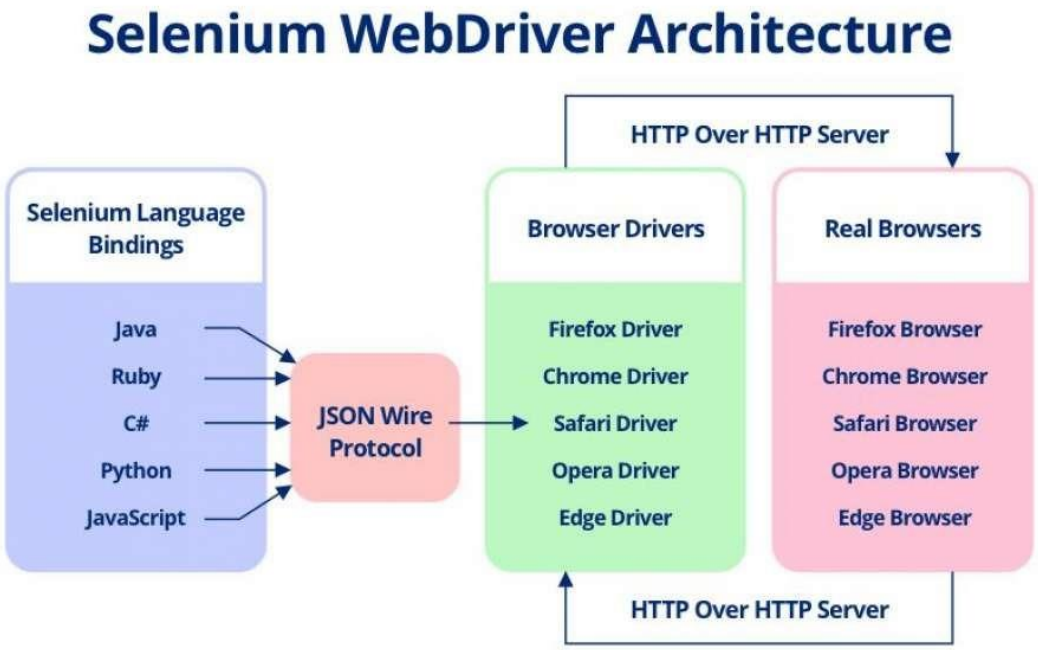
Browsers drivers, tarayıcıları otomatik olarak açıp kapatmak, sayfaları yüklemek, tarayıcı ayarlarını yapılandırmak gibi görevleri gerçekleştirir. Ayrıca, tarayıcıların farklı özelliklerini ve davranışlarını kontrol etmek için çeşitli yöntemler sağlar. Bu

sayede tarayıcının boyutunu değiştirmek, JavaScript etkinleştirmek, devre dışı bırakmak, çerezleri yönetmek gibi işlemler yapılabilmektedir.

Browsers drivers, otomasyon testlerinin tarayıcılarda güvenilir bir şekilde çalışabilmesini sağlar. Bu sürücüler, test senaryolarının tarayıcılar arasında taşınabilirliğini sağlar ve birden fazla tarayıcıda testlerin kolayca yürütülmesini mümkün kılar.

Bölüm 2

2.1 ACUnit



Şekil 1.5 – Selenium Diyagram (Kaynak: https://hackr.io/blog/what-is-selenium-webdriver)

ACUnit , OOP(Object-Orianted Programming) kullanılarak Visual Studio üzerinden C# dilinde yazılmış bir test otomasyon kütüphanesidir. POM(Page Object Modal) metodolojisi kullanılarak geliştirildiği için otomasyon içerisinde geliştirme yapmak isteyen kullanıcılara rahatlık sağlamaktadır. Selenium IDE si çalışma mantığı esas alınarak desteklenen fonksyionlar kullanıcı dostu metotlara dönüştürülüp kod bilgisini minimum seviyeye indirerek yazılım test otomasyonları hazırlamayı hedefler.

ACUnit'in başlıca özellikleri şunlardır:

1. Kayıt ve Oynatma: kullanıcının bir tarayıcı üzerinde yaptığı etkileşimleri kaydedebilir ve bu etkileşimleri otomatik olarak oynatabilir. Bu, test senaryolarını hızlı bir şekilde oluşturmanızı ve tekrar kullanmanızı sağlar.

2. Kolay Kullanıcı Arayüzü: Kullanıcı dostu bir arayüz sunar. Senaryoları kaydetmek, düzenlemek ve yönetmek için kod yazmanıza gerek kalmadan test senaryolarını oluşturmanıza olanak tanır.

3. Nesne Tanımlama ve Kayıt: Manuel nesne tanımlama ve kayıt özelliğine sahiptir. Bu özellik, uygulamanın kullanıcı arayüzündeki öğeleri tanımlamanızı ve test senaryolarınızı daha hızlı bir şekilde oluşturmanızı sağlar.

4. Açık dünya Özelliği: Open source kod yapısıyla Dünya üzerindeki farklı kullanıcılar tarafından düzenlenebilir. Kullanıcılar kütüphane içerisindeki metotları kullanarak kendi kütüphanelerini de geliştirebilirler.

5. OOP yapısı: OOP geliştirilen kod yapısıyla kod tekrarından kaçınılan kullanıcı dostu bir deneyim sağlar.

2.2 Projede Kullanılan Kütüphaneler

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Interactions;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;
```

Şekil 1.6 – Kullanılan Kütüphaneler

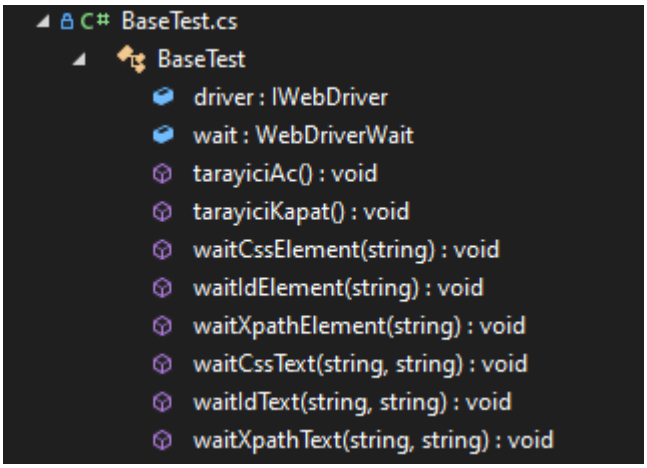
Proje oluşturulurken Selenium IDE sinin en temel fonksiyonları esas alınmıştır. Proje içerisinde kullanılan kütüphanelerin sürümler:

Kütüphane	Versiyon
DotNetSeleniumExtras.WaitHelpers	3.11.0
Microsoft.NET.Test.Sdk -	17.5.0
MSTest.TestAdapter	3.0.2
MSTest.TestFrameWork	3.0.2
NUnit	3.13.2
NUnit.ConsoleRunner	3.16.3
Selenium.Support	4.9.0
Selenium.WebDriver	4.9.0
Selenium.WebDriver.ChromeDriver	113.0.5672.6300
Selenium.WebDriver.IEDriver	4.8.1
System.Threading	4.3.0

Tablo 1.1 – Kullanılan Kütüphaneler tablosu

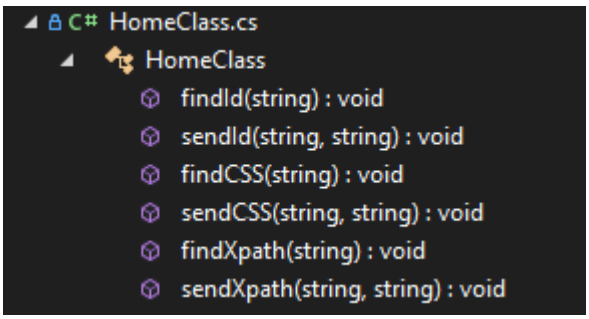
2.3 Proje Yapısı

Proje Visual Studio üzerinden Class Library yapısıyla oluşturuldu. Visual Studio'nun kendi içerisinde UnitTest proje çeşitleri de bulunmaktadır fakat global bir kütüphane oluşturabilmek için ClassLibrary yapısı kuruldu. Proje içerisinde şu anda 3 farklı sınıf bulunmaktadır.

1- 

Şekil 1.7 – BaseTest Metotları

BaseTest: Kütüphane içerisindeki en temel fonksiyonları barındıran bir sınıftır. Tarayıcıyı açma , kapama ve elementler arasındaki bekleme sürelerini barındırır. Diğer sınıflara kalıtım ile bağlanmıştır [5],[6].

2- 

Şekil 1.8 – HomeClass Metotları

HomeClass: Otomasyonda kullanılacak Metotları içerisinde bulunduran bir sınıftır. Elementleri yakalama , tıklama , yazı gönderme gibi Metotlara sahiptir. Şu anda birden fazla element çeşidini yakalayabilmektedir. Sınıf geliştirilmeye devam edilmektedir [5],[6].

3-


```
[TestFixture]

public class MainTest:HomeClass
{
    [Test]

    public void LoginTest()
    {

        findCSS("div.modal-close");
        findId("onetrust-accept-btn-handler");
        findCSS("div.link.account-user");
        sendId("login-email", "ahmet");
        sendId("login-password-input", "testsifre");
        findXpath("//html/body/div[1]/div[3]/div[3]/div[1]/form/button");
        Thread.Sleep(2000);

    }

    [Test]

    public void HelpinTest()
    {
        findCSS("div.modal-close");
        findId("onetrust-accept-btn-handler");
        findXpath("//div[@id='headerMain']/div/div/ul/li[3]/a");
        Thread.Sleep(2000);
        findCSS("div.layout-wrapper>div:nth-of-type(13)>span");
        Thread.Sleep(2000);
    }
```

Şekil 1.9 – Main test örnekleri

MainTest: Otomasyona örnek olması açısından eklenmiş bir sınıftır. Kullanıcı uzunca kod yazma gereği duymadan hazır fonksiyonlar ile sadece elementleri değişken olarak göstererek test otomasyonunu koşabilir [5],[6].

2.4 Proje Hedefi

Proje içerisinde kullanılan sınıf ve metotlar temel seviyededir. Proje tesliminden sonra geliştirilmeye devam edilecektir. Selenium'un desteklediği ve otomasyon dünyasının ihtiyacı olan bütün sınıf ve Metotlar eklendikten sonra kütüphane yayınlanacaktır. Kullanılan kütüphaneler en stabil versiyonlara sahip olmasına rağmen sürekli geliştirilen ve güncellenen bir yapıda olmak durumundadır. Kütüphane yayınlandıktan sonra arayüze implemente edilecektir. Arayüz konusunda uygulama, browser ya da browser extension olarak çalışmalar yapılmaktadır [7].

Kaynaklar

[1] Microsoft Learn: https://learn.microsoft.com/tr-tr/visualstudio/get-started/visual-studio-ide?view=vs-2022

[2] Appium: https://medium.com/automationmaster/appium-mobile-app-automation-tutorial-2-527d6d78998a

[3] Cucumber: https://www.youtube.com/watch?v=X0MdVNiina4

[4] Katalon Studio: https://katalon.com/resources-center/blog/easy-automation-testing

[5] SeleniumHQ Documentation: https://www.selenium.dev/documentation/en/

[6] Selenium WebDriver C# Tutorial: https://www.guru99.com/all-about-selenium-with-c-sharp.html

[7] Selenium Getting Started with Selenium C#: https://www.pluralsight.com/guides/getting-started-with-selenium-in-csharp

[8] with C# - Complete Tutorial: https://www.toolsqa.com/selenium-c-sharp/

Ekler

```
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Interactions;
using OpenQA.Selenium.Support.UI;
using SeleniumExtras.WaitHelpers;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace AhmetSeleniumProje
{
   public class BaseTest
   {
      public IWebDriver driver;
      public WebDriverWait wait;


      [SetUp]
      public void tarayiciAc()
      {
         driver = new ChromeDriver();
         wait = new WebDriverWait(driver, TimeSpan.FromSeconds(5));
         driver.Manage().Timeouts().ImplicitWait = TimeSpan.FromSeconds(10);
         driver.Manage().Window.Maximize();
         driver.Navigate().GoToUrl("https://www.trendyol.com");
      }

      [TearDown]
      public void tarayiciKapat()
      {
         driver.Quit();
      }


      public void waitCssElement(string elementCss)
      {

         wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.CssSelector(elementCss)));
      }


      public void waitIdElement(string elementId)
      {
         wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.Id(elementId)));
```

```
        }

        public void waitXpathElement(string elementxPath)
        {
            wait.Until(ExpectedConditions.VisibilityOfAllElementsLocatedBy(By.XPath(elementxPath)));
        }

        public void waitCssText(string elementCss, string elementText)
        {
            wait.Until(ExpectedConditions.TextToBePresentInElementLocated(By.Id(elementCss),
elementText));
        }


        public void waitIdText(string elementId, string elementText)
        {
            wait.Until(ExpectedConditions.TextToBePresentInElementLocated(By.Id(elementId),
elementText));
        }
        public void waitXpathText(string elementXpath, string elementText)
        {
            wait.Until(ExpectedConditions.TextToBePresentInElementLocated(By.XPath(elementXpath),
elementText));
        }



    }
}
```

```
using OpenQA.Selenium;
using OpenQA.Selenium.Interactions;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Threading.Tasks;

namespace AhmetSeleniumProje
{
    public class HomeClass:BaseTest
    {
        public void findId(string elementId)
        {
            driver.FindElement(By.Id(elementId)).Click();
        }

        public void sendId(string elementId, string sendText)
        {
            driver.FindElement(By.Id(elementId)).SendKeys(sendText);
        }
        public void findCSS(string elementCss)
        {
            driver.FindElement(By.CssSelector(elementCss)).Click();
        }
        public void sendCSS(string elementCss, string sendText)
        {
            driver.FindElement(By.CssSelector(elementCss)).SendKeys(sendText);
        }
```

```
        public void findXpath(string elementXpath)
        {
            driver.FindElement(By.XPath(elementXpath)).Click();
        }
        public void sendXpath(string elementXpath, string sendText)
        {
            driver.FindElement(By.XPath(elementXpath)).SendKeys(sendText);
        }

    }
}
```

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using System;
using System.Threading;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium;
using OpenQA.Selenium.Support.UI;
using OpenQA.Selenium.Support.Extensions;
using NUnit.Compatibility;
using NUnit;
using NUnit.Framework;
using SeleniumExtras.WaitHelpers;

namespace AhmetSeleniumProje
{
    [TestFixture]
    public class MainTest:HomeClass
    {
        [Test]
        public void LoginTest()
        {

            findCSS("div.modal-close");
            findId("onetrust-accept-btn-handler");
            findCSS("div.link.account-user");
            sendId("login-email", "ahmet");
            sendId("login-password-input", "testsifre");
            findXpath("//html/body/div[1]/div[3]/div[3]/div[1]/form/button");
            Thread.Sleep(2000);

        }

        [Test]
        public void HelpinTest()
        {
            findCSS("div.modal-close");
            findId("onetrust-accept-btn-handler");
            findXpath("//div[@id='headerMain']/div/div/ul/li[3]/a");
            Thread.Sleep(2000);
            findCSS("div.layout-wrapper>div:nth-of-type(13)>span");
            Thread.Sleep(2000);
        }

        [Test]
        public void Deneme()
        {

            waitCssElement("div.modal-close");
```

```
            findCSS("div.modal-close");
            waitIdText("onetrust-accept-btn-handler", "KABUL ET");
            findId("onetrust-accept-btn-handler");


        }
    }
}
```

Özgeçmiş

Adı Soyadı: Ahmet Ali Can

Eğitim:

2015–2020 Karabük Üniversitesi , Mekatronik Mühendisliği

2021–2023 İzmir Kâtip Çelebi Üniversitesi, Lisansüstü Yazılım Müh.

İş Deneyimi:

2020 – 2021 Üç Boyutlu Yazılım Bilgi ve İletişim Sistemleri

2021 – 2022 A101

2022 – 2023 Logo Yazılım