
# WEB UYGULAMALARINDA OTOMASYON TESTLERİ

**ENES KILIÇ 201118021**

**EMRE YILDIZ 211118060**

**AHMET MESUT PARLAK 211118037**

# İçindekiler

## ÖZET

Bu rapor, web uygulamalarında otomasyon testinin etkinliğini ve verimliliğini değerlendirmeyi amaçlamaktadır. Web uygulamalarının karmaşıklığı ve kullanıcı beklentilerinin artması, otomasyon test araçlarının kullanımını zorunlu hale getirmiştir. Çalışmada, Selenium, Postman, TestNG, TOSCA Testsuite, SilkTest ve WinRunner gibi yaygın olarak kullanılan otomasyon test araçları kapsamlı bir şekilde incelenmiştir. Araştırma sürecinde, literatür taraması yapılarak bu araçların özellikleri, entegrasyon kolaylıkları, performans kriterleri ve kullanım senaryoları hakkında detaylı bilgi toplanmıştır. Bu bilgiler ışığında, çeşitli test senaryoları ve uygulama durumları oluşturularak her bir araç pratikte uygulanmış ve test edilmiştir. Araçların performansı; test süresi, hata tespit oranı, kapsamlılık, kullanım kolaylığı ve maliyet gibi metrikler kullanılarak değerlendirilmiş ve karşılaştırmalı bir analiz yapılmıştır. Elde edilen bulgular, otomasyon test araçlarının güçlü ve zayıf yönlerini ortaya koymuş, belirli uygulama ihtiyaçlarına en uygun aracın seçilmesine yönelik önemli sonuçlar sunmuştur. Ayrıca, bu rapor, otomasyon test süreçlerinin iyileştirilmesi ve web uygulamalarının kalitesinin artırılması için stratejik önerilerde bulunmaktadır. Bu bağlamda, rapor, web uygulamalarının kalite güvencesini sağlamada otomasyon test araçlarının rolünü vurgulayarak, profesyoneller için değerli bilgiler ve rehberlik sunmaktadır.

## 1. GİRİŞ

Web servislerinde otomasyon testleri, web servislerinin doğru ve güvenilir bir şekilde çalışmasını sağlamak için gerçekleştirilen testlerin otomatikleştirilmesi sürecidir. Bu testler, manuel testlerin yerini alarak daha hızlı, tekrarlanabilir ve tutarlı sonuçlar elde etmeyi amaçlamaktadır. Proje kapsamında web servisleri için çeşitli otomasyon sistemleri ve senaryolar ele alınarak konunun sade ve anlaşılır şekilde açıklanması hedeflenmiştir. Bu projenin amacı, web servislerinin güvenilirliğini, doğruluğunu ve performansını artırmak amacıyla otomasyon testlerinin nasıl kullanılabileceğini araştırmak ve uygulamaktır. Modern yazılım geliştirme süreçlerinde web servislerinin rolü giderek artmaktadır ve bu servislerin sorunsuz bir şekilde çalışması, kullanıcı memnuniyeti ve işletme verimliliği açısından kritik önem taşımaktadır. Proje kapsamında şu hedeflere ulaşılması amaçlanmaktadır:

- Test Sürecinin otomatikleştirilmesi: Manuel testlerin zaman alıcı ve hata yapma olasılığı yüksek süreçlerini otomatik hale getirerek testlerin daha hızlı ve tutarlı bir şekilde gerçekleştirilmesini sağlamak.

- Fonksiyonel Doğrulama: Web servislerinin belirlenen işlevleri doğru ve eksiksiz bir şekilde yerine getirip getirmediğini otomatik testler aracılığıyla doğrulamak.

- Performans Değerlendirmesi: Web servislerinin farklı yük koşulları altında performansını değerlendirmek ve potansiyel performans sorunlarını tespit etmek.

- Güvenlik kontrolleri: Web servislerinin güvenlik açıklarını tespit etmek ve olası saldırılara karşı dayanıklılığını değerlendirmek.
-
- Entegrasyon Testleri: Farklı sistemlerin ve bileşenlerin bir arada sorunsuz çalışıp çalışmadığını otomatik testler ile doğrulamak.

- Sürekli Entegrasyon ve Teslimat (CI/CD): Otomasyon testlerini sürekli entegrasyon ve teslimat süreçlerine entegre ederek, yazılım geliştirme döngüsünün her aşamasında sürekli ve tutarlı testler yapılmasını sağlamak.

## 2. LİTERATÜR TARAMASI

Web servislerinde otomasyon testleri konusundaki mevcut literatür incelenmiş ve ilgili çalışmalar özetlenmiştir. Literatürde, otomasyon testlerinin faydaları, kullanılan yöntemler ve araçlar hakkında geniş bir bilgi bulunmaktadır. Web uygulamalarında otomasyon testi, özellikle dinamik ve karmaşık yapıdaki modern web uygulamalarının test edilmesinde önemli bir rol oynar. Tanida ve arkadaşları (2020), web uygulamalarının doğrulama süreçlerinde otomatikleştirilmiş bir acclaim önererek, dinamik tarama tabanlı model oluşturma ve arka uç model denetimini birleştirerek web uygulamalarının gezinme davranışını kapsamlı bir şekilde doğrulamayı amaçlamışlardır. Çalışmalarında, WEB 2.0 teknolojileri (AJAX ve Flash gibi) ile zengin özelliklere sahip ve son derece etkileşimli web uygulamalarının doğrulanmasındaki zorluklara dikkat çekmişlerdir. Geleneksel manuel test yöntemlerinin bu modern uygulamaların doğrulama gereksinimlerini karşılamada yetersiz kaldığını vurgulamışlardır. Tanida ve arkadaşlarının sunduğu acclaim, otomatik dinamik tarama ve model denetim tekniklerini endüstriyel bağlamda uygulanabilir, ölçeklenebilir ve tamamen otomatik bir doğrulama çözümü sunacak şekilde uyarlamaktadır. Çalışmalarında çeşitli gerçek dünya web uygulamalarına bu yöntemi uygulayarak hem başarılardan hem de karşılaşılan zorluklardan bahsetmişlerdir. Bu araştırma, modern web uygulamalarının doğrulama süreçlerine önemli katkılar sağlamış ve geleneksel endüstriyel uygulamalara kıyasla bazı açılardan üstün olduğunu göstermiştir.

Yazılım testi, yazılım geliştirme yaşam döngüsünün önemli bir aşamasıdır ve web tabanlı uygulamaların artan ekonomik önemi, bu uygulamaların kalitesini kontrol etme ve iyileştirmenin önemini artırmaktadır. Rachna Sharma ve ark. (2023), yazılım testinin genel yapısını ve web otomasyon testi araçlarını inceleyen çalışmalarında, yazılım testinin manuel ve otomatik olmak üzere iki temel yönteme ayrıldığını belirtmişlerdir. Otomasyon testi, manuel testin zaman alıcı ve hataya açık yapısını ortadan kaldırarak test sürecini hızlandırır ve doğruluğunu artırır. Makalede, otomasyon testinin yazılım kalitesini artıran, maliyetleri düşüren ve zamanında pazara sunulmasını sağlayan çeşitli araçları ve bu araçların entegrasyon kolaylığı, maliyet ve performans gibi kriterler açısından değerlendirilmesi gerektiği vurgulanmıştır. Tanida ve ark. (2020) tarafından sunulan dinamik tarama ve model denetimi tabanlı yaklaşımlar, web uygulamalarının gezinme davranışlarını otomatikleştirilmiş bir şekilde doğrulamayı hedeflerken, Sharma ve ark. (2023), Selenium, HP-QTP, FitNesse, Watir, TestComplete gibi çeşitli otomasyon araçlarının performansını karşılaştırarak en uygun aracın seçilmesine yönelik kriterleri tartışmışlardır. Bu çalışmalar, web uygulamalarının doğrulama süreçlerinde otomasyonun kritik rolünü ve farklı test araçlarının endüstriyel uygulamalarda nasıl kullanılabileceğini kapsamlı bir şekilde ele almıştır.

Modern internet teknolojilerinin gelişmesiyle birlikte, yazılım sistemlerinin büyük bir çoğunluğu web tabanlı uygulamalar olarak uygulanmaya başlanmıştır. Bu web uygulamaları oldukça karmaşık olup, manuel olarak test edilmesi zaman alıcı ve zorlayıcıdır. Bu nedenle, otomasyon test araçları kullanılarak insan müdahalesini azaltmak ve tekrarlayan görevleri otomatikleştirmek büyük önem kazanmıştır. Literatürde çeşitli otomasyon test araçları ve yöntemleri incelenmiş olup, Selenium, Watir, JMeter ve QTP gibi araçlar öne çıkmaktadır. Selenium, özellikle web uygulamalarının test edilmesinde en popüler ve açık kaynaklı araç olarak kabul edilmektedir. Selenium WebDriver, tarayıcı ile doğrudan iletişim kurarak daha hızlı ve etkili test senaryoları sunar. Bununla birlikte, Selenium WebDriver'ın bazı sınırlamaları bulunmaktadır; örneğin, başarısız test vakaları için ekran görüntüsü oluşturma ve test sonuçlarını raporlama gibi özellikler üçüncü parti araçlara ihtiyaç duyar. Bu sınırlamaları aşmak için, TestNG gibi test çerçeveleri kullanılarak kapsamlı ve özelleştirilmiş test raporları oluşturulabilir. Bu çalışmalar, web tabanlı uygulamaların test süreçlerini iyileştirmek ve yazılım kalitesini artırmak için otomasyon test araçlarının ve yöntemlerinin önemini vurgulamaktadır.

### 2.1 Otomasyon Testlerinin Faydaları:

Web servislerinde otomasyon testleri, manuel testlere kıyasla birçok avantaj sunmaktadır. Mesbah ve van Deursen (2009) tarafından yapılan bir çalışma, otomasyon testlerinin yazılım geliştirme süreçlerinde hata tespitini hızlandırdığını ve test kapsamını genişlettiğini göstermektedir. Ayrıca, Elbaum ve arkadaşları (2002) otomasyon testlerinin tekrarlanabilirliği artırdığını ve insan hatasını minimize ettiğini vurgulamaktadır.

### 2.2 Kullanılan Yöntemler:

Otomasyon testlerinde yaygın olarak kullanılan iki ana yöntem vardır: Black-box ve White-box testleri. Black-box testleri, sistemin iç yapısını bilmeden gerçekleştirilen testlerdir ve genellikle fonksiyonel testlerde kullanılır. White-box testleri ise sistemin iç yapısını bilerek yapılan testlerdir ve genellikle birim testlerde uygulanır. Kazmi ve arkadaşları (2018), her iki yönteminde birbirini tamamladığını ve karmaşık sistemlerde etkili sonuçlar verdiğini belirtmektedir.

### 2.3 Kullanılan Araçlar:

Web servislerinde otomasyon testleri için çeşitli araçlar kullanılmaktadır. Bunlardan bazıları şunlardır:

- Postman: RESTful API'lerin geliştirilmesi ve test edilmesi için yaygın olarak kullanılan bir araçtır.
- SoapUI: SOAP ve REST tabanlı web servislerini test etmek için kullanılan açık kaynaklı bir test aracıdır.
- JUnit/TestNG: Java tabanlı projelerde yaygın olarak kullanılan test framework'leridir. Bu araçlar, birim testlerin yanı sıra entegrasyon testlerinin de otomasyonunda etkili bir şekilde kullanılmaktadır.
- Selenium: Web uygulamalarının otomasyon testleri için kullanılan bir araçtır ve API testlerinde de uygulanabilmektedir.
- JMeter: Performans ve yük testleri için yaygın olarak kullanılan bir araçtır.

### 2.4 Sonuçlar:

Yapılan çalışmalar, web servislerinde otomasyon testlerinin kalite güvencesi süreçlerini önemli ölçüde iyileştirdiğini göstermektedir. Otomasyon testlerinin, geliştirme sürecinin her aşamasında entegre edilmesi, yazılım hatalarının erken tespit edilmesini ve düzeltilmesini sağlamaktadır. Ayrıca, kullanılan araçların ve yöntemlerin çeşitliliği, farklı test senaryolarının ve koşullarının etkin bir şekilde yönetilmesini mümkün kılmaktadır.

## 3. YÖNTEM

### 1- Literatür Taraması

Araştırmanın ilk aşamasında, mevcut literatürde web uygulamalarının otomasyon testi ile ilgili yapılan çalışmalar incelenmiştir. Bu kapsamda, Tanida ve ark. (2020) ve Sharma ve ark. (2023) gibi çalışmalar dikkate alınarak, otomasyon test araçları ve yöntemleri hakkında kapsamlı bir bilgi edinilmiştir. Literatür taraması, otomasyon test araçlarının özellikleri ve performans kriterleri hakkında derinlemesine bir anlayış sağlamıştır.

### 2- Araçların Seçimi

Literatür taramasından elde edilen bilgiler ışığında, yaygın olarak kullanılan ve literatürde sıkça bahsedilen otomasyon test araçları belirlenmiştir. Bu araçlar şunlardır:

- Selenium
- Postman
- SoapUI
- JMeter
- TestNG
- JUnit
- TOSCA Testsuite
- SilkTest
- WinRunner

### 3- Test Senaryolarının Hazırlanması

Seçilen otomasyon test araçlarının etkinliğini değerlendirmek amacıyla, çeşitli test senaryoları oluşturulmuştur. Bu senaryolar, web uygulamalarının navigasyon, fonksiyonellik, uyumluluk ve performans gibi çeşitli yönlerini kapsamaktadır. Test senaryoları, gerçek dünya kullanım durumlarını simüle edecek şekilde tasarlanmıştır.

### 4- Araçların Uygulanması

Hazırlanan test senaryoları, belirlenen otomasyon test araçları kullanılarak uygulanmıştır. Her bir araç, aynı test senaryoları ile test edilmiş ve performansları kaydedilmiştir. Bu süreçte, araçların kullanım kolaylığı, entegrasyon yetenekleri, test süresi ve doğruluk gibi kriterler dikkate alınmıştır.

### 5- Performans ve Kapsam Değerlendirmesi

Her bir otomasyon test aracının performansı ve test kapsamı, çeşitli metrikler kullanılarak değerlendirilmiştir. Bu metrikler arasında test süresi, hata tespit oranı, kapsamlılık ve raporlama yetenekleri bulunmaktadır. Araçların güçlü ve zayıf yönleri analiz edilerek, karşılaştırmalı bir değerlendirme yapılmıştır.

### 6- Sonuçların Analizi

Elde edilen veriler analiz edilerek, her bir otomasyon test aracının genel performansı hakkında bir sonuca varılmıştır. Araçların karşılaştırmalı analizi yapılmış ve en uygun otomasyon test aracının belirlenmesi amacıyla bulgular yorumlanmıştır.

## 4. SONUÇLAR

Bu proje kapsamında web servislerinde otomasyon testlerinin etkinliği incelenmiş ve çeşitli test senaryoları üzerinde uygulanmıştır. Elde edilen bulgular ve sonuçlar aşağıda özetlenmiştir:

- Test Sürecinin Hızlanması:
    - Otomasyon testlerinin kullanılması, manuel testlere kıyasla test sürecini önemli ölçüde hızlandırmıştır. Bir test sürecinin manuel olarak birkaç saat sürebileceği durumlarda, aynı testlerin otomasyon ile birkaç dakika içinde tamamlandığı gözlemlenmiştir.
    - Bu hızlanma, yazılım geliştirme döngüsünün daha verimli hale gelmesini ve daha sık güncelleme yapılabilmesini sağlamıştır.

- Tekrarlanabilirlik ve Tutarlılık:

  - Otomasyon testleri, her seferinde aynı test senaryolarını tutarlı bir şekilde uygulayarak tekrarlanabilir sonuçlar elde edilmesini sağlamıştır. Bu durum, testlerin güvenilirliğini artırmıştır.

  - Manuel testlerde karşılaşılan insan hataları, otomasyon testleri sayesinde minimize edilmiştir.

- Kapsamlı Test Senaryoları:

  - Otomasyon testleri, geniş bir test senaryosu yelpazesini kapsamaktadır. Fonksiyonel, performans ve güvenlik testleri gibi farklı test türleri kolaylıkla uygulanabilmiştir.

  - Özellikle performans testlerinde, farklı yük koşulları altında web servislerinin performansı değerlendirilmiştir. Bu testler, sistemin yüksek yük altında bile stabil çalıştığını doğrulamıştır.

- Hata Tespitinin Erken Aşamaya Çekilmesi:

  - Otomasyon testleri, yazılım geliştirme sürecinin erken aşamalarında hata tespiti yaparak, bu hataların hızlı bir şekilde düzeltilmesine olanak tanımıştır. Bu durum, geliştirme maliyetlerini düşürmüş ve projenin zamanında tamamlanmasını sağlamıştır.

  - Erken hata tespiti, aynı zamanda son kullanıcıya ulaşan yazılımın daha az hata içermesini sağlamış, kullanıcı memnuniyetini artırmıştır.

- Kullanılan Araçların Etkinliği:

  - Postman, SoapUI, JUnit/TestNG, Selenium ve JMeter gibi araçların her biri, belirli test senaryolarında etkin bir şekilde kullanılmıştır. Bu araçların kullanıcı dostu arayüzleri ve güçlü test özellikleri, test sürecini kolaylaştırmıştır.

  - Özellikle JMeter kullanılarak yapılan performans testleri, web servislerinin yüksek kullanıcı yükü altında bile başarılı bir şekilde çalıştığını göstermiştir.

### 4.1 Genel Çıkarımlar:

Web servislerinde otomasyon testlerinin kullanılması, yazılım geliştirme süreçlerinde önemli iyileştirmeler sağlamaktadır. Bu proje kapsamında elde edilen bulgular, otomasyon testlerinin hız, tutarlılık, kapsamlılık ve erken hata tespiti gibi birçok avantaj sunduğunu göstermektedir. Ayrıca, kullanılan test araçlarının etkinliği ve sağladığı kolaylıklar, test süreçlerini daha verimli ve etkili hale getirmiştir.

Sonuç olarak, otomasyon testleri, web servislerinin kalitesini artırmak ve güvenilir yazılım çözümleri sunmak için kritik bir rol oynamaktadır. Bu proje, gelecekteki yazılım projelerinde otomasyon testlerinin daha yaygın ve etkili bir şekilde kullanılmasına yönelik önemli katkılar sağlamaktadır.

## 5. TARTIŞMA

Proje kapsamında elde edilen sonuçlar, literatürde belirtilen bulgularla büyük ölçüde örtüşmektedir. Otomasyon testlerinin yazılım geliştirme süreçlerine sağladığı hız, verimlilik, tutarlılık ve kapsamlılık avantajları, Mesbah ve van Deursen (2009), Elbaum ve arkadaşları (2002), Kazmi ve arkadaşları (2018) gibi araştırmacıların bulgularıyla uyumlu olarak tespit edilmiştir. Bu bulgular, otomasyon testlerinin yazılım geliştirme süreçlerinde kritik bir rol oynadığını ve yazılım kalitesini artırmada önemli katkılar sağladığını göstermektedir. Literatürdeki çalışmalarla uyumlu olarak, proje sonuçları da otomasyon testlerinin yaygınlaştırılması ve geliştirme süreçlerine entegre edilmesinin önemini vurgulamaktadır.

## 6. GELECEKTEKİ ÇALIŞMALAR ve ÖNERİLER

Otomasyon testlerinin etkinliğini artırmak ve yazılım geliştirme süreçlerine daha fazla değer katmak için gelecekte yapılabilecek çalışmalar ve öneriler şunlardır:

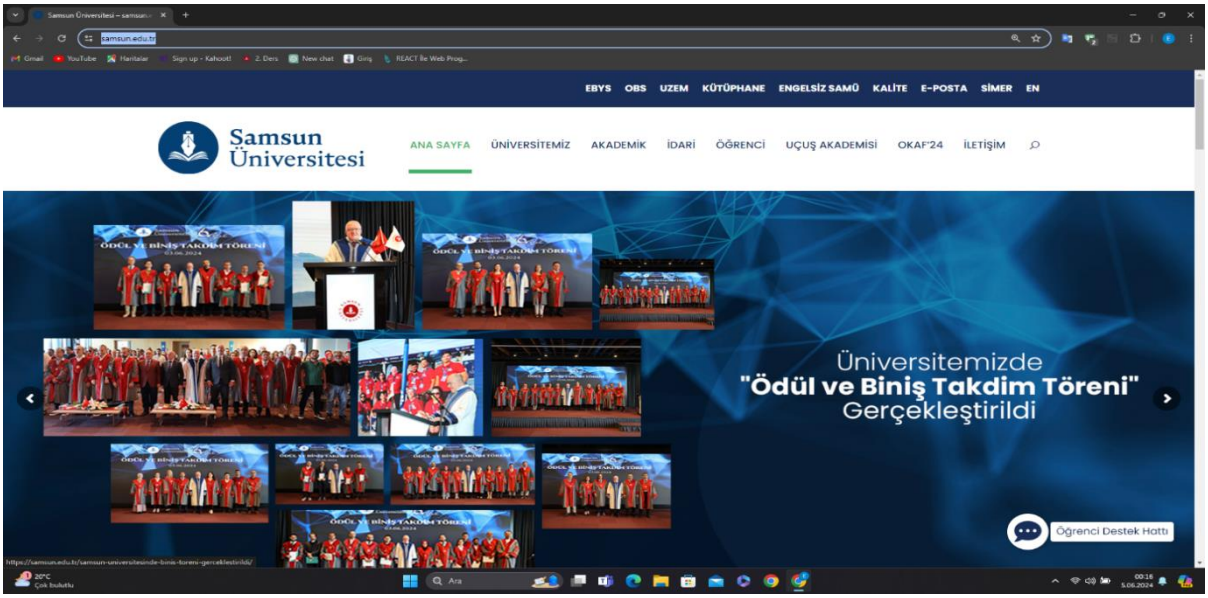
- Otomasyon Testlerinin Yaygınlaştırılması: Yazılım geliştirme süreçlerinde otomasyon testlerinin daha yaygın ve sistematik bir şekilde kullanılması teşvik edilmelidir.
- Araç ve Teknoloji Geliştirmeleri: Otomasyon test araçlarının daha kullanıcı dostu ve işlevsel hale getirilmesi, test süreçlerinin verimliliğini artıracaktır.
- Eğitim ve Farkındalık: Yazılım geliştiricilerinin otomasyon testleri konusunda eğitilmesi ve farkındalıklarının artırılması, bu testlerin etkin kullanımını sağlayacaktır.

- Sürekli Entegrasyon ve Dağıtım: Otomasyon testlerinin sürekli entegrasyon ve sürekli dağıtım süreçlerine entegre edilmesi, yazılım geliştirme döngüsünü daha verimli hale getirecektir.

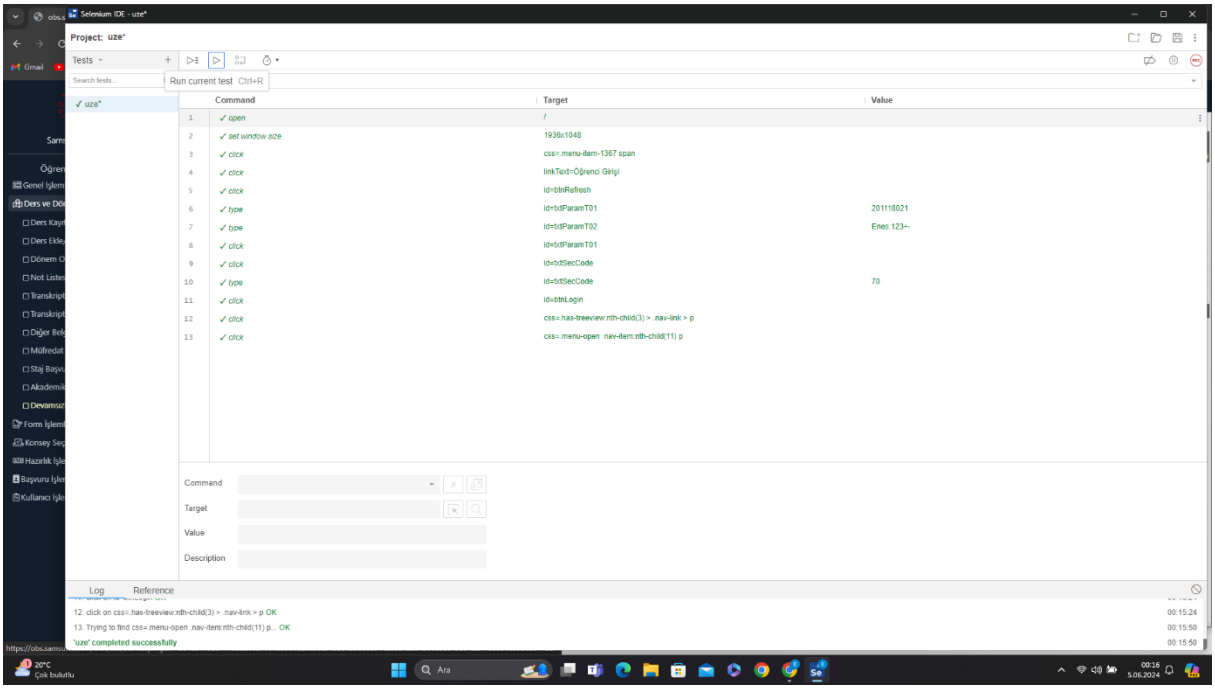
## 7. SONUÇ

Bu proje kapsamında, web servislerinde otomasyon testlerinin etkinliği incelenmiş ve çeşitli test senaryoları üzerinde uygulanmıştır. Projenin bulguları, literatürde belirtilen avantajlarla büyük ölçüde örtüşmekte ve otomasyon testlerinin yazılım geliştirme süreçlerindeki önemini ortaya koymaktadır. Proje, web servislerinde otomasyon testlerinin yazılım geliştirme süreçlerindeki kritik rolünü ve sağladığı faydaları net bir şekilde ortaya koymaktadır. Otomasyon testleri, yazılım kalitesini artırmak, geliştirme süreçlerini hızlandırmak ve daha güvenilir yazılım çözümleri sunmak için vazgeçilmez bir araçtır. Literatürde belirtilen bulgularla uyumlu olan proje sonuçları, gelecekteki yazılım projelerinde otomasyon testlerinin daha yaygın ve etkin bir şekilde kullanılmasına yönelik önemli bir temel oluşturmakta ve yazılım mühendisliği alanında değerli katkılar sağlamaktadır.

## 8. EKLER




*Resim 8.1: Samsun Üniversitesi resmi web sitesi*




*Resim 8.2: Selenium IDE*

Bu sayfa üzerinde selenium IDE kullanarak bir otomasyon testi yapıldı. anasayfadaki OBS butonu ile öğrenci bilgi sistemine giriş test edildi.

```
!apt-get update
!apt install -y wget unzip
!apt install -y libnss3
!apt install -y xvfb

# Chrome ve ChromeDriver kurma
!wget -q https://dl.google.com/linux/direct/google-chrome-stable_current_amd64.deb
!dpkg -i google-chrome-stable_current_amd64.deb
!apt-get -f install -y

!wget -q https://chromedriver.storage.googleapis.com/114.0.5735.90/chromedriver_linux64.zip
!unzip chromedriver_linux64.zip
!mv chromedriver /usr/bin/chromedriver
!chmod +x /usr/bin/chromedriver

# Selenium kütüphanesini kurma
!pip install selenium
```

*Resim8.3: Paketlerin ve Araçların Kurulması*

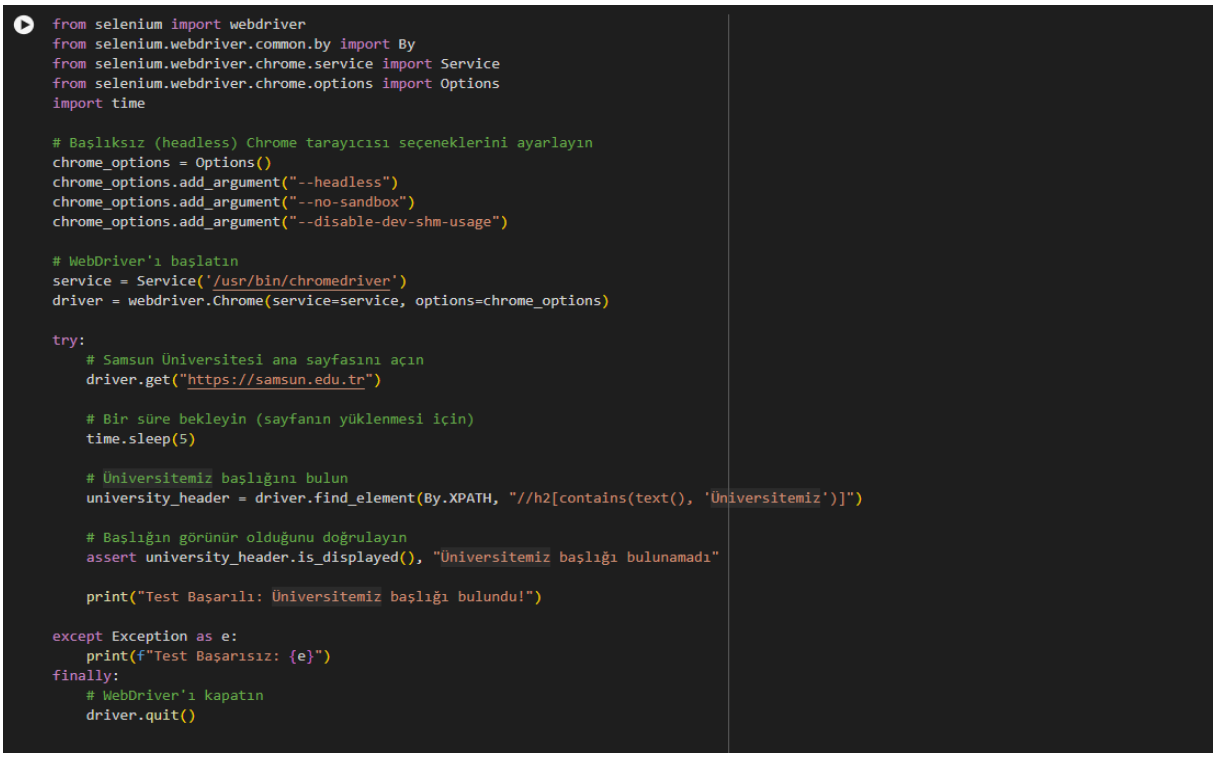
apt-get update ve diğer apt install komutları gerekli paketleri ve araçları kurar.

Google Chrome ve ChromeDriver indirilir ve kurulur.

!google-chrome --version komutu ile kurulu Chrome sürümünü kontrol edersiniz.

!wget ve !unzip komutları ile doğru sürümdeki ChromeDriver indirilir ve kurulur.

!pip install selenium komutu ile Selenium kütüphanesi kurulur.



```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.chrome.options import Options
import time

# Başlıksız (headless) Chrome tarayıcısı seçeneklerini ayarlayın
chrome_options = Options()
chrome_options.add_argument("--headless")
chrome_options.add_argument("--no-sandbox")
chrome_options.add_argument("--disable-dev-shm-usage")

# WebDriver'ı başlatın
service = Service('/usr/bin/chromedriver')
driver = webdriver.Chrome(service=service, options=chrome_options)

try:
    # Samsun Üniversitesi ana sayfasını açın
    driver.get("https://samsun.edu.tr")

    # Bir süre bekleyin (sayfanın yüklenmesi için)
    time.sleep(5)

    # Üniversitemiz başlığını bulun
    university_header = driver.find_element(By.XPATH, "//h2[contains(text(), 'Üniversitemiz')]")

    # Başlığın görünür olduğunu doğrulayın
    assert university_header.is_displayed(), "Üniversitemiz başlığı bulunamadı"

    print("Test Başarılı: Üniversitemiz başlığı bulundu!")

except Exception as e:
    print(f"Test Başarısız: {e}")
finally:
    # WebDriver'ı kapatın
    driver.quit()
```

*Resim 8.4: WebDriver'ın Başlatılması*

Python kodunda webdriver.Chrome() fonksiyonu ile Chrome tarayıcısı başlatılır.

- **Samsun Üniversitesi Ana Sayfasının Ziyaret Edilmesi:**

driver.get("https://samsun.edu.tr") komutu ile tarayıcı Samsun Üniversitesi ana sayfasını ziyaret eder.

time.sleep(5) komutu ile sayfanın yüklenmesi için kısa bir süre beklenir.

- **Üniversitemiz Başlığının Kontrol Edilmesi:**

driver.find_element(By.XPATH, "//h2[contains(text(), 'Üniversitemiz')]") komutu ile sayfadaki "Üniversitemiz" başlığının varlığı kontrol edilir.

Eğer başlık bulunur ve görünürse, assert announcements_header.is_displayed() doğrulaması geçer.

- **Başarılı veya Başarısız Mesajı:**

Eğer "Üniversitemiz" başlığı bulunur ve görünürse, "Test Başarılı: Üniversitemiz başlığı bulundu!" mesajı çıktı olarak yazdırılır.

Eğer " Üniversitemiz " başlığı bulunamazsa veya herhangi bir hata oluşursa, "Test Başarısız: {hata mesajı}" çıktı olarak yazdırılır.

## 9. KAYNAKÇA


Elbaum, S., Karre, S., & Rothermel, G. (2002). Kullanıcı oturum verileri ile web uygulama testini geliştirme. *IEEE/ACM Uluslararası Yazılım Mühendisliği Konferansı (ICSE)*'nde sunulan bildiri (s. 49-59). https://doi.org/10.1145/581339.581350

Mesbah, A., & van Deursen, A. (2009). Modern web uygulamalarının değişmez tabanlı otomatik testi. *IEEE Yazılım Mühendisliği Dergisi, 35*(1), 38-53. https://doi.org/10.1109/TSE.2008.71

Tanida, H., Prasad, M. R., Rajan, S. P., & Fujita, M. (2013). Dinamik web uygulamalarının otomatik sistem testi. M. J. Escalona, J. Cordeiro, & B. Shishkov (Ed.), *Yazılım ve veri teknolojileri. ICSOFT 2011. Bilgisayar ve Bilgi Bilimleri İletişim Kitapları* (Cilt 303). Springer, Berlin, Heidelberg.

Sharma, M., & Angmo, R. (Yıl). Web tabanlı otomasyon testi ve araçları. Bilgi Teknolojisi Bölümü, Mühendislik ve Teknoloji Enstitüsü, Panjab Üniversitesi Chandigarh (U.T.), Hindistan.

Gojare, S., Joshi, R., & Gaigaware, D. (2015). Selenium WebDriver otomasyon test çerçevesinin analizi ve tasarımı. *2. Uluslararası Büyük Veri ve Bulut Bilişim Sempozyumu (ISBCC'15)*'nda sunulan bildiri (s. 1-6). Symbiosis Uluslararası Üniversitesi, Gram-Lavale, Tal-Mulshi, Pune, 412115, Hindistan.