



Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ
2024-2025 EĞİTİM ÖĞRETİM YILI GÜZ
DÖNEMİ

Yazılım Kalite Güvence Temelleri

DÖNEM PROJESİ RAPORU

Bilgisayar Mühendisliği Bölümü

Murat Can Kargı – 171421011

Selin Nisa Çolak – 170421007

DANIŞMAN

Öğr. Gör. Hilal RAKICI

İSTANBUL, 2025

1. Uygulama ve API Analizi

Çalışma kapsamında test süreci, kullanıcı arayüzü (UI) ve uygulama programlama arayüzü (API) içeren bir demo platform olan ToolsQA web sitesi üzerinde gerçekleştirilmiştir. İlgili uygulama, <https://demoqa.com/> adresinden erişilebilen, otomasyon testleri ve QA (Quality Assurance) eğitimleri için hazırlanmış, çeşitli kullanıcı senaryolarını içeren interaktif bir örnek platformdur.

1.1. Uygulamanın Genel Yapısı

ToolsQA uygulaması, kullanıcıların web UI elementleri üzerinde test senaryoları çalıştırılabilmesine olanak sağlayacak şekilde yapılandırılmıştır. Site, aşağıdaki ana bileşenleri içermektedir:

- Elements: Butonlar, metin kutuları, checkbox'lar, radio button'lar gibi temel HTML elementlerinin test edilebildiği bölümdür.
- Forms: Kullanıcıdan veri toplanmasını simüle eden form yapıları içerir. Özellikle “Practice Form” sayfası, form doğrulama testlerinin yapılması için kullanılmaktadır.
- Alerts, Frames & Windows: Uyarı kutuları, sekme geçişleri ve iframe etkileşimleri gibi kullanıcı deneyimini etkileyen yapılar yer almaktadır.
- Widgets: Takvim, slider, progress bar gibi gelişmiş bileşenler test edilmektedir.
- Interactions: Sürükle-bırak, seçim kutuları ve sıralama işlemleri gibi kullanıcı etkileşimlerinin analiz edildiği birimdir.
- Book Store Application: Login sistemi içeren ve kitap ekleme, silme, listeleme gibi işlemlerin yapılabilindiği, aynı zamanda REST API üzerinden testlerin de gerçekleştirilebildiği bir mini uygulamadır.

1.2. API İncelemesi: Book Store Application

Uygulama bünyesindeki Book Store Application modülü, kullanıcı doğrulama, kitap listeleme, kitap kaydetme ve silme gibi işlemler için bir RESTful API sunmaktadır. Testler sırasında bu modül öncelikli olarak analiz edilmiştir.

API uç noktaları aşağıdaki işlemleri desteklemektedir:

- Login (POST /Account/v1/Login): Kullanıcı kimlik doğrulaması sağlar.
- Generate Token (POST /Account/v1/GenerateToken): JWT token üretimi için kullanılır.
- Get Books (GET /BookStore/v1/Books): Kitap listesini döndürür.

- Add Book (POST /BookStore/v1/Books): Kullanıcı hesabına kitap ekler.
- Delete Book (DELETE /BookStore/v1/Book): Kullanıcının seçili kitabını siler.
- Delete All Books (DELETE /BookStore/v1/Books): Tüm kitapları siler.

API tarafında yapılan testlerde Postman kullanılmış, her bir uç nokta için HTTP istekleri yapılandırılarak başarılı ve başarısız senaryolar gözlemlenmiştir. Token bazlı doğrulama sistemi test edilmiştir.

2. Oluşturulan Test Senaryoları

2.1. API Testleri

Bu çalışma kapsamında, API uç noktalarının işlevselliğini ve güvenilirliğini değerlendirmek amacıyla çeşitli test senaryoları oluşturulmuştur. Testler, Postman aracı kullanılarak manuel olarak yapılandırılmış ve otomatik test komutları aracılığıyla gerçekleştirilmiştir. Aşağıda, her bir uç nokta için uygulanan testlerin ayrıntıları sunulmuştur.

2.1.1. Kullanıcı Oluşturma Testi (POST /Account/v1/User)

Bu uç noktaya gönderilen istek ile yeni bir kullanıcı oluşturulması hedeflenmiştir. Testler aşağıdaki amaçlara yönelik olarak gerçekleştirilmiştir:

- İsteğe karşılık olarak HTTP 201 Created durum kodunun dönüp dönmediği kontrol edilmiştir.
- Dönen yanıt içerisinde userID ve username alanlarının bulunup bulunmadığı doğrulanmıştır.
- Yanıt içeriğinde herhangi bir şekilde password bilgisinin yer almadığı test edilmiştir.
- Aynı kullanıcı adıyla yapılan tekrar isteklerde HTTP 406 Not Acceptable durum kodunun dönmesi beklenmiş ve bu durum senaryosu test edilmiştir.

2.1.2. Token Oluşturma Testi (POST /Account/v1/GenerateToken)

Bu test ile kullanıcı adı ve şifre bilgileri kullanılarak bir erişim token'ı üretilip üretilmediği test edilmiştir. Gerçekleştirilen kontroller aşağıdaki gibidir:

- İsteğe karşılık olarak HTTP 200 OK durum kodunun dönmesi beklenmiş ve doğrulanmıştır.
- Dönen JSON yanıtında status alanının bulunduğu ve bu değer "Success" ya da "Failed" olması gerektiği test edilmiştir.
- Eğer token başarıyla oluşturulmuşsa (status === "Success"), token değeri koleksiyon değişkenlerine atanmıştır.
- expires alanının status === "Success" durumunda boş olmaması; status === "Failed"

durumunda ise boş olması gerektiği kontrol edilmiştir.

- Yanıt içeriğinde bulunan result alanının, duruma uygun bir mesaj içerip içermediği test edilmiştir: "User authorized successfully" veya "User authorization failed".

2.1.3. Kitap Listeleme Testi (GET /BookStore/v1/Books)

Bu test, sistemde yer alan kitapların listelenip listelenmediğini denetlemek amacıyla uygulanmıştır. Aşağıdaki doğrulamalar gerçekleştirilmiştir:

- İsteğe karşılık olarak HTTP 200 OK durum kodunun dönmesi beklenmiştir.
- Yanıtın JSON formatında olup olmadığı kontrol edilmiştir.
- Yanıt gövdesinin boş olmaması beklenmiş ve bu durum test edilmiştir.
- Yanıtın books adlı bir dizi (array) içerip içermediği değerlendirilmiştir.
- En az bir kitabın listelenmiş olması gerektiği varsayımıyla, books dizisinin uzunluğunun sıfırdan büyük olması beklenmiştir.
- Ayrıca, yanıt süresinin 80ms altında kalması durumunun performans kriteri olarak test edilmesi sağlanmıştır.

2.2. Selenium Testleri

Uygulamanın kullanıcı arayüzü (UI) katmanında doğru şekilde çalışıp çalışmadığını sınamak amacıyla Selenium WebDriver kullanılarak otomasyon testleri uygulanmıştır. Söz konusu testler, Chrome tarayıcısı üzerinde çalıştırılmış olup, ChromeDriverManager aracılığıyla tarayıcı sürücüsü kurulmuştur. Gerçekleştirilen testler dört temel senaryo etrafında yapılandırılmıştır: kullanıcı kaydı (sign-up), giriş yapma (login), tekil kitap silme ve tüm kitapların silinmesi. Her bir test senaryosu aşağıda detaylı bir biçimde açıklanmıştır.

2.2.1. Kullanıcı Kayıt (Sign-Up) Testi

Kayıt ekranının işlevselliğini sınamak amacıyla <https://demoqa.com/register> adresine yönlendirme yapılmıştır. Gerekli kullanıcı bilgileri (ad, soyad, kullanıcı adı, şifre) form alanlarına girilmiştir. reCAPTCHA doğrulamasının iframe içinde yer aldığı tespit edilmiş ve bu iframe'e geçiş yapılarak doğrulama kutucuğu işaretlenmiştir. Daha sonra "Register" butonuna tıklanarak kayıt işlemi başlatılmıştır.

Test sonucunda ekran görüntüsü alınmış ve başarılı kayıt mesajının DOM üzerinde görünür hâle gelip gelmediği kontrol edilmiştir. Eğer kayıt başarıyla gerçekleştirilmişse, mesajın içeriği konsola yazdırılmış; aksi durumda işlem başarısız olarak değerlendirilmiştir. Bu test ile birlikte, form bileşenlerinin erişilebilirliği, CAPTCHA entegrasyonunun uyumluluğu ve kayıt sonrası geri bildirimin görüntülenebilirliği test edilmiştir.

2.2.2. Kullanıcı Giriş (Login) Testi

Giriş işleminin test edilmesi amacıyla <https://demoqa.com/login> adresine erişim sağlanmıştır. Kullanıcı adı ve şifre bilgileri girilmiş; ardından "Login" butonuna tıklanarak oturum açma süreci başlatılmıştır.

İşlem sonrasında kullanıcı adı bilgisinin görüntülendiği ögenin DOM'da yer alıp almadığı kontrol edilmiştir. Bu doğrulama ile, oturum açma işleminin başarıyla gerçekleştirilip gerçekleştirilmediği test edilmiştir. Test süreci boyunca iki farklı zaman aralığında ekran görüntüsü alınarak görsel doğrulama yapılmıştır.

2.2.3. Tekil Kitap Silme Testi

Bu senaryoda, önceden oturum açmış bir kullanıcı üzerinden kitap silme işlemi gerçekleştirilmiştir. <https://demoqa.com/login> adresine erişilmiş ve kullanıcı bilgileri girilerek giriş yapılmıştır. Ardından, kitap listesinde yer alan belirli bir kitabın silinmesi hedeflenmiştir.

"Delete" butonuna tıklanmış ve açılan onay penceresinde "OK" butonuna basılarak işlem tamamlanmıştır. Silme işleminin ardından ekran görüntüsü alınmış, silme işlemi sırasında oluşabilecek olası hatalar try-except bloğu kullanılarak loglanmıştır. Bu test aracılığıyla, UI üzerinde silme işlevinin hem görsel olarak hem de fonksiyonel açıdan çalışıp çalışmadığı gözlemlenmiştir.

2.2.4. Tüm Kitapları Silme Testi

Son senaryo kapsamında, giriş yapmış bir kullanıcının kitaplıkta yer alan tüm kitapları silme işlemi test edilmiştir. "Delete All Books" butonuna tıklanmış ve sonrasında gelen uyarı penceresindeki "OK" butonuna basılarak tüm verilerin silinmesi sağlanmıştır. İşlem sonucunda, kullanıcı arayüzünde kitapların temizlendiği doğrulanmak üzere ekran görüntüsü alınmış ve işlem süresince oluşabilecek olası hatalar kaydedilmiştir.

Bu testle birlikte, çoklu veri silme işlemlerinin kullanıcıya sunulan arayüz üzerinden sorunsuz bir şekilde gerçekleştirilip gerçekleştirilmediği test edilmiştir.

2.2.5. Form Doldurma ve Gönderme Testi

Sistem üzerindeki formların kullanıcı girişiyle nasıl etkileşim kurduğu değerlendirilmek amacıyla, <https://demoqa.com/automation-practice-form> adresine erişim sağlanmıştır.

Sayfa yüklendikten sonra, test sürecinin sağlıklı yürütülebilmesi için sayfa alt kısmında yer alan sabit reklam bileşenleri JavaScript komutları aracılığıyla sayfa DOM'undan kaldırılmıştır. Form alanları sırasıyla doldurulmuş; ad, soyad ve e-posta bilgileri ilgili alanlara girilmiştir. Cinsiyet seçimi kısmında "Female" seçeneği işaretlenmiş, telefon numarası alanına örnek bir numara girilmiştir. Konu olarak "Math" yazılmış ve önerilen seçeneklerden biri seçilmiştir. İlgi alanları kapsamında "Sports" ve "Music" kutucukları işaretlenmiş, ardından açık adres bilgisi girilmiştir. Eyalet ve şehir seçimleri sırasıyla "NCR" ve "Delhi" olarak belirlenmiş ve scroll işlemi ile seçim alanlarına erişim sağlanmıştır. Form gönderme işlemi tamamlandıktan sonra ekran görüntüsü alınmış ve formun başarıyla iletildiği doğrulanmıştır. İşlem sonunda tarayıcı oturumu sonlandırılmıştır.

3. API ve UI Test Sonuçları

3.1. API Test Sonuçları

API testleri, sistemin arka plan servislerinin doğruluğunu, güvenilirliğini ve hata toleransını değerlendirmek amacıyla yürütülmüştür. Testler sırasında kullanıcı kaydı, oturum açma, veri listeleme ve kitap silme gibi işlevler ele alınmıştır.

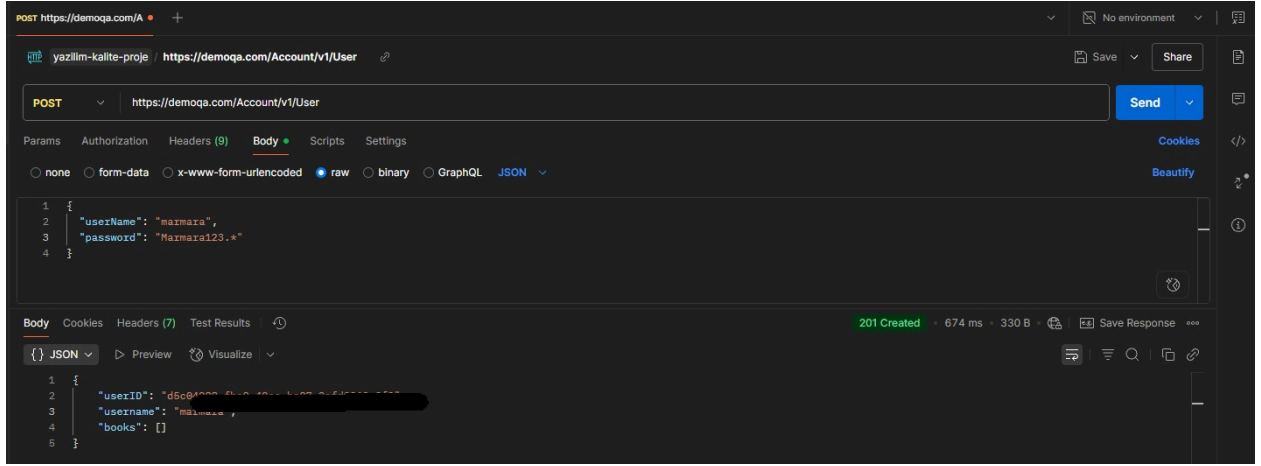
Her bir API isteği Postman aracı kullanılarak gönderilmiş ve gelen yanıtlar HTTP durum kodlarına göre değerlendirilmiştir. Gerçekleştirilen testler sonucunda, sistemin hem API hem de kullanıcı arayüzü (UI) üzerinden çeşitli işlemleri genel olarak doğru şekilde yürüttüğü gözlemlenmiştir. Ancak, API testleri sırasında bazı uç noktalardan dönen 200 ve 400 HTTP durum kodları incelendiğinde, belirli senaryolarda beklenmeyen davranışlar ve hata mesajları tespit edilmiştir. Bu durum, hata yönetimi mekanizmalarının bazı durumlarda yetersiz kaldığını göstermektedir. Testlerde aşağıdaki bulgular öne çıkmıştır:

- Başarılı kullanıcı kaydı sonucunda sistem tarafından geçerli bir token üretilmiştir.
- Giriş işlemleri sonucunda kullanıcı bilgileri başarılı bir şekilde döndürülmüş ve oturum doğrulanmıştır.
- Yetkisiz erişim denemelerinde sistemin uygun güvenlik yanıtları verdiği belirlenmiştir.

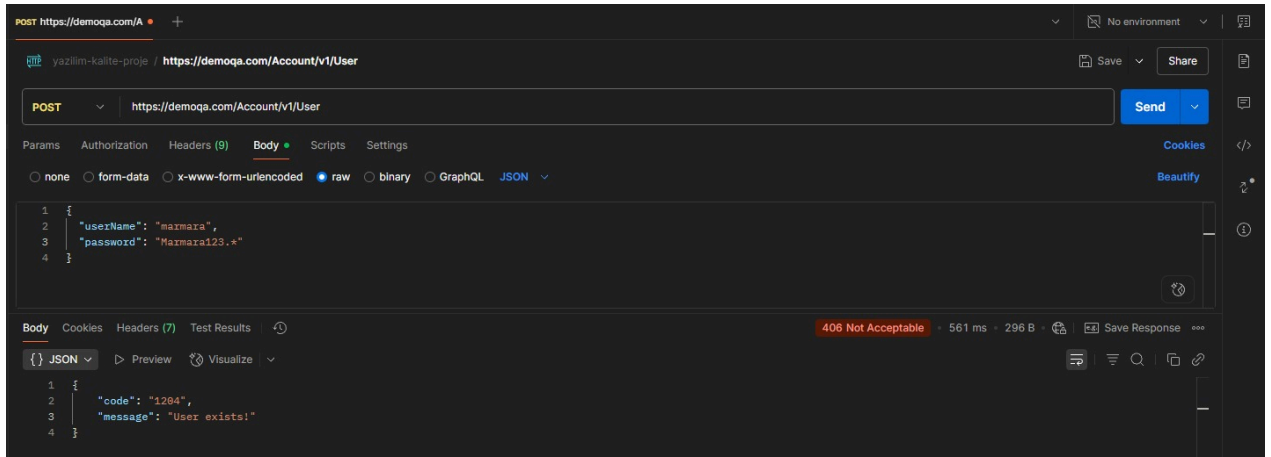
Bu doğrultuda, API katmanının genel olarak işlevsel ve güvenilir olduğu sonucuna varılmıştır.

/Account/v1/User = Bu API, kullanıcı kaydı (kullanıcı adı ve şifre oluşturma) işlemlerini gerçekleştirir. Kullanıcıların sisteme yeni hesap oluşturmalarını sağlar.

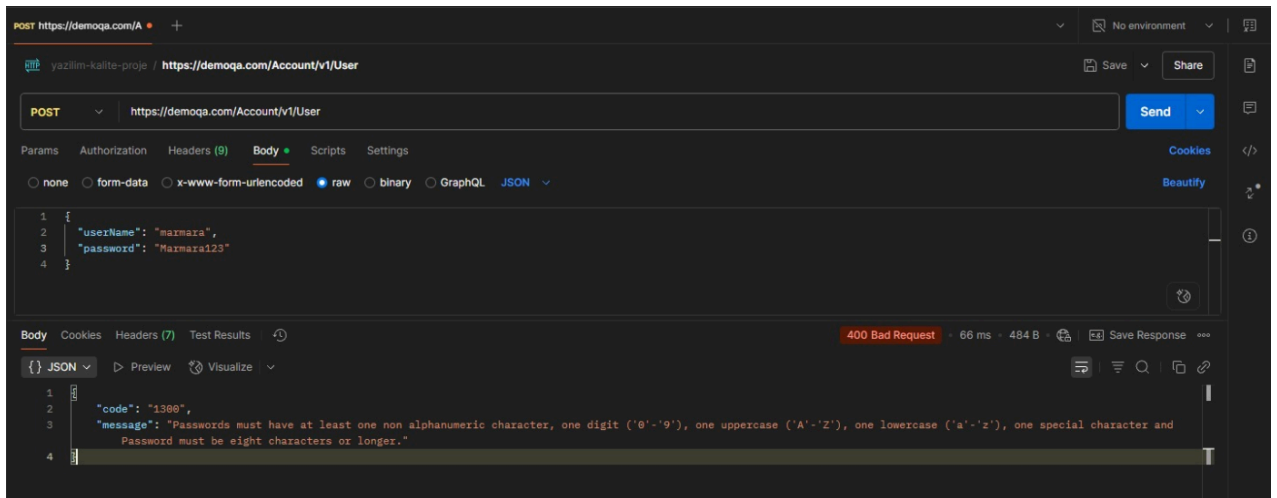
Yeni bir kullanıcı oluşturulurken tüm bilgilerin doğru girildiği senaryo (201 Created) :



Daha önceden oluşturulmuş kullanıcının tekrar oluşturulmaya çalışıldığı senaryo (406 Not Acceptable):

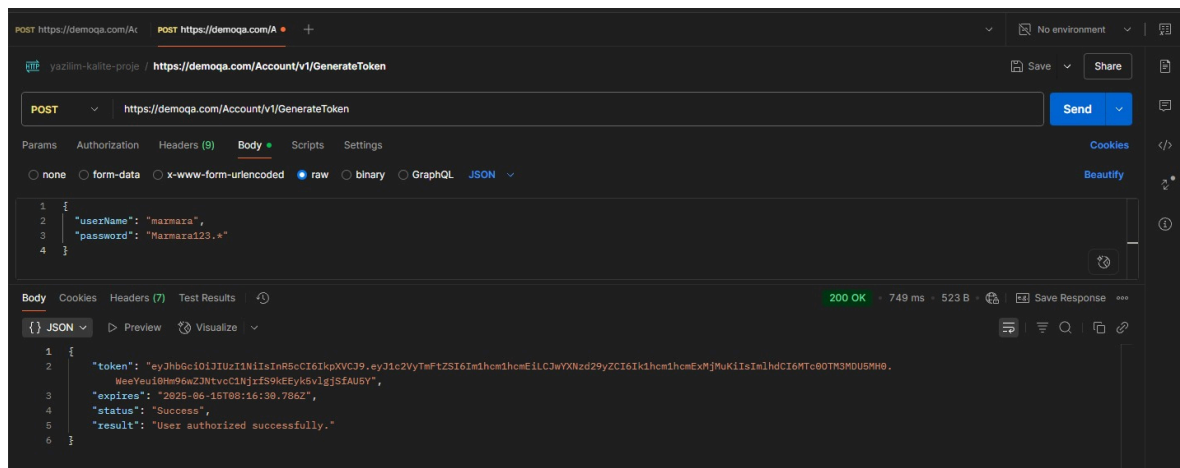


Şifre oluşturulurken istenilen durumların karşılanmadığı senaryo (400 Bad Request):



A screenshot of the Swagger client interface. At the top, the URL bar shows 'https://demoqa.com/Account/v1/User'. The 'Body' tab is selected, and the message 'This request does not have a body' is displayed. Below, the 'Body' section shows a JSON response: { "code": "1200", "message": "UserName and Password required." }. The status bar at the bottom indicates a '400 Bad Request' with a response time of 182 ms and a size of 312 B.

Kullanıcı bilgilerinin doğru girildiği senaryo (200 OK):



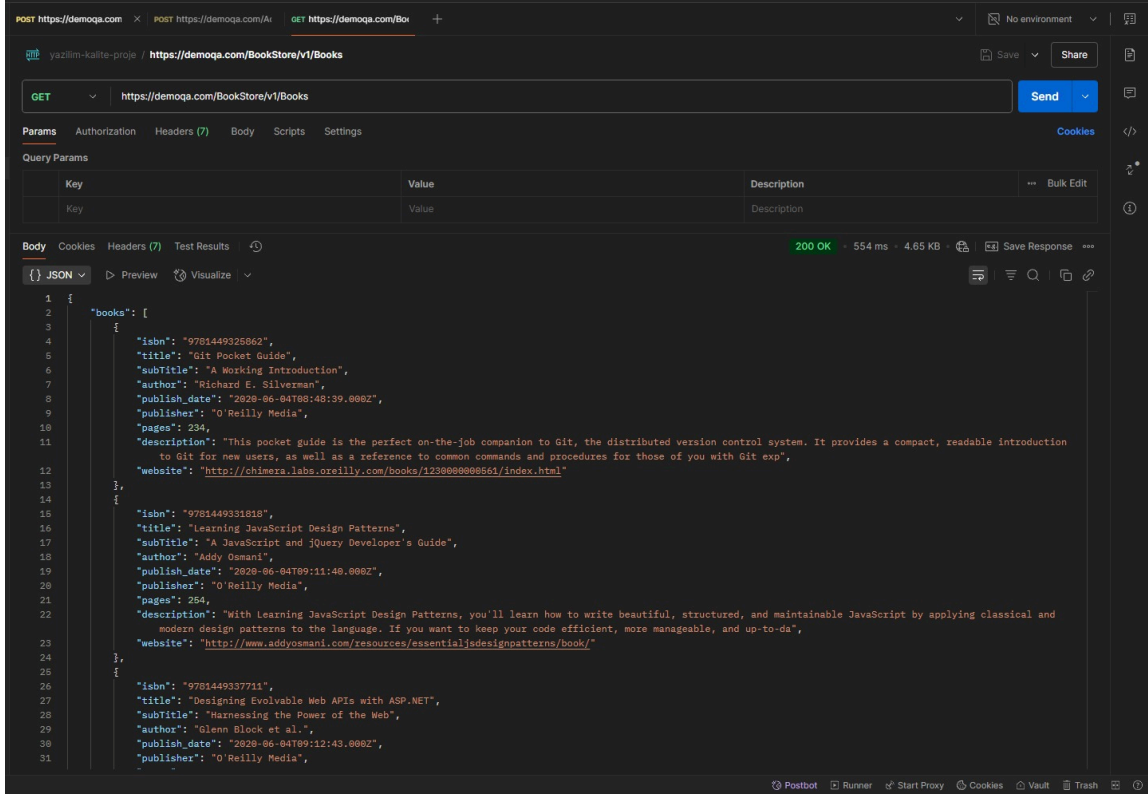
The screenshot shows a REST client interface with the following details:

- URL:** `https://demoqa.com/Account/v1/GenerateToken`
- Method:** `POST`
- Body Type:** `raw` (selected)
- Request Body (raw):**

```
{
  "userName": "marmar1",
  "password": "Marmar123.*"
}
```
- Response:** `200 OK` (803 ms, 329 B)
- Response Body (JSON):**

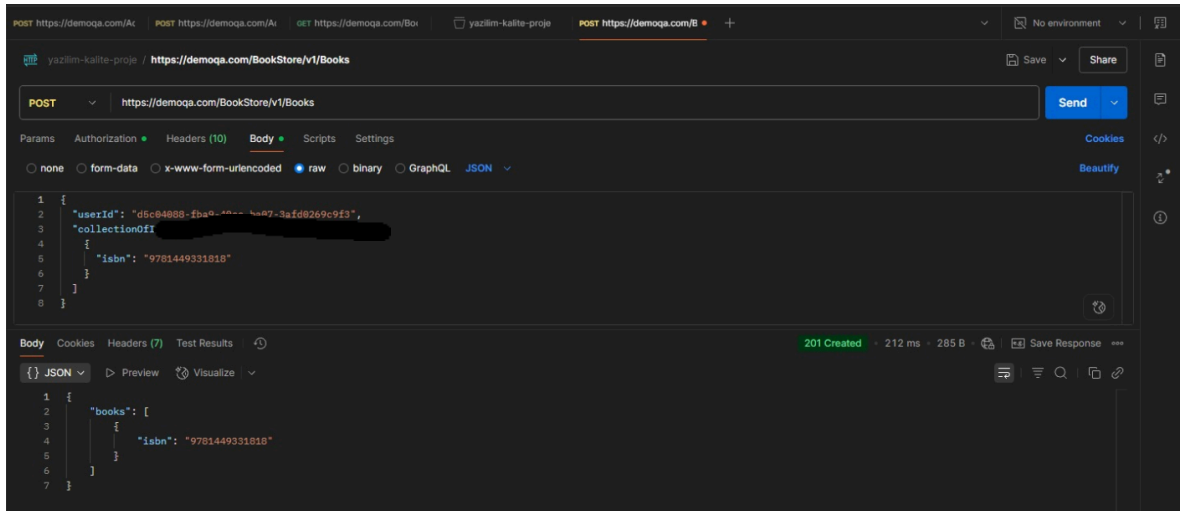
```
{
  "token": null,
  "expires": null,
  "status": "Failed",
  "result": "User authorization failed."
}
```


GET /BookStore/v1/Books = Bu API, kitap listesini döner. Kitapların detaylarını almak için kullanılır.

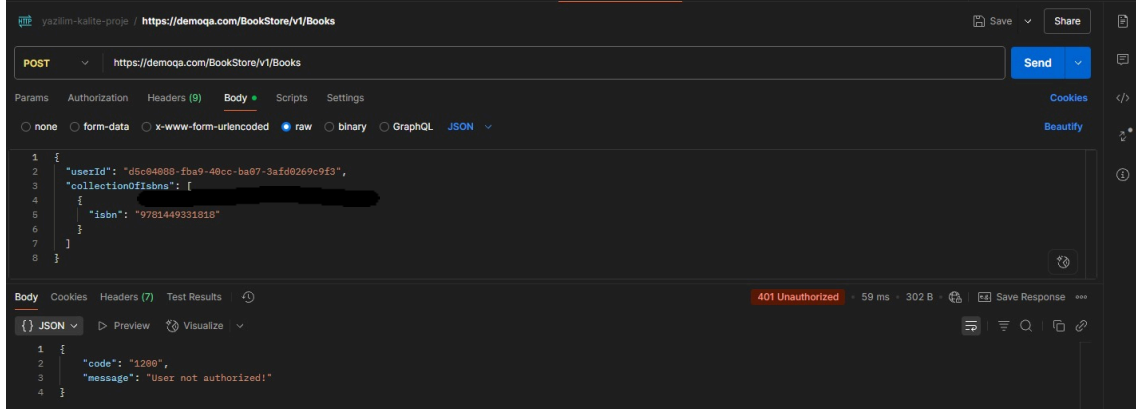


POST /BookStore/v1/Books = Bu API, kullanıcıya kitap eklemek için kullanılır. Kullanıcının kitap koleksiyonuna bir ISBN'e göre kitap ekler.

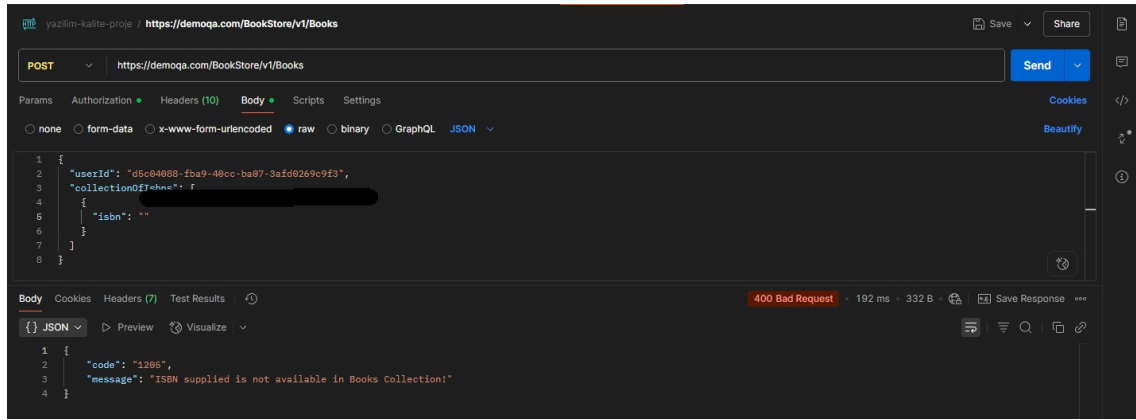
Tüm bilgilerin (token, userId, isbn) doğru girildiği senaryo (201 Created):



Authentication bilgilerinin hatalı girildiği veya girilmediği senaryo (401 Unauthorized):

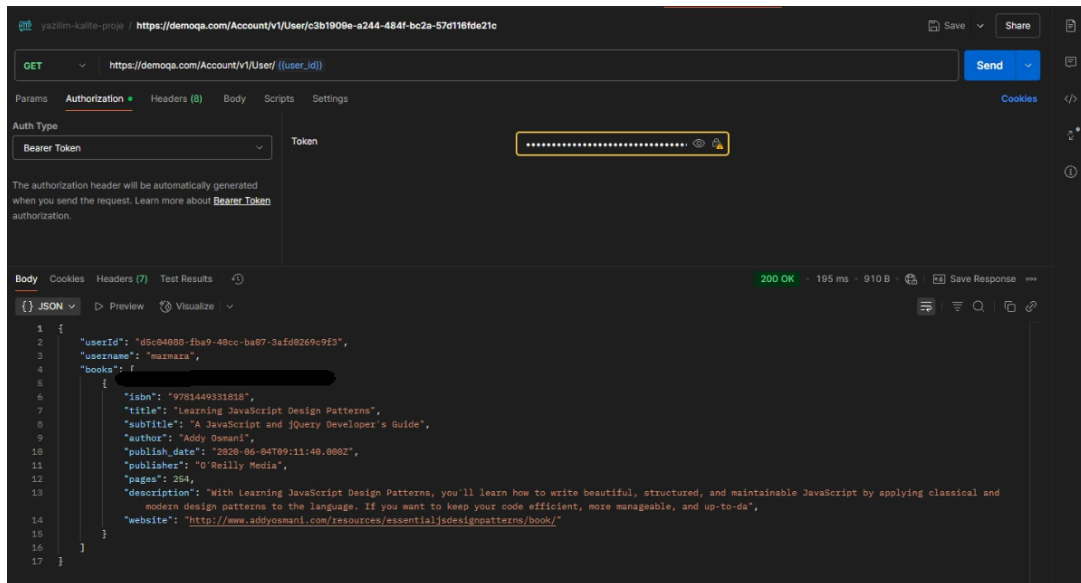


Kitabın ISBN numarasının yanlış girildiği veya girilmediği senaryo



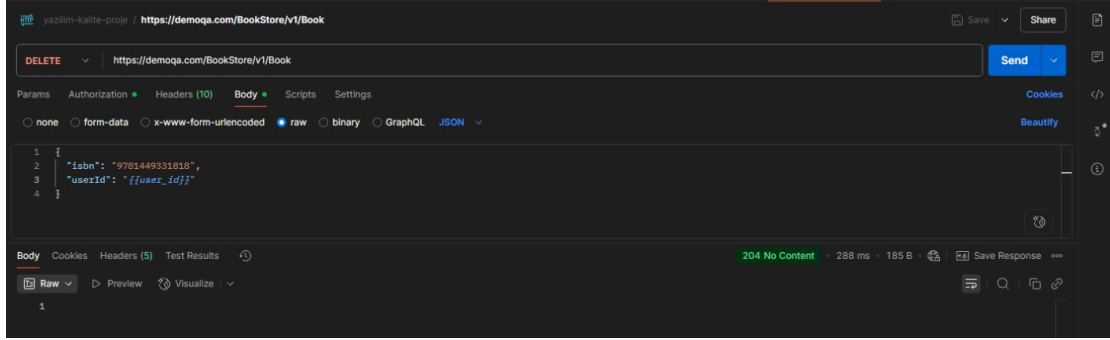
/Account/v1/User/{{user_id}}= Belirtilen kullanıcı ID'sine sahip kullanıcıya ait kitaplar listelenir.

Authentication bilgilerinin doğru girildiği senaryo (200 OK):

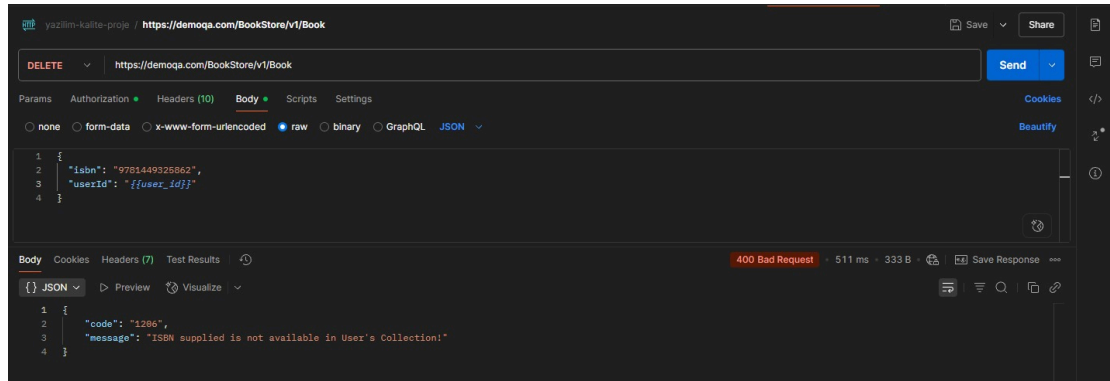


DELETE /BookStore/v1/Book = Belirtilen kullanıcıdan bir kitabı silmek için kullanılır. Kullanıcının kitap koleksiyonundan seçilen ISBN'li kitabı kaldırır.

Tüm bilgilerin(token, isbn, userId) doğru girildiği senaryo (204 No Content):

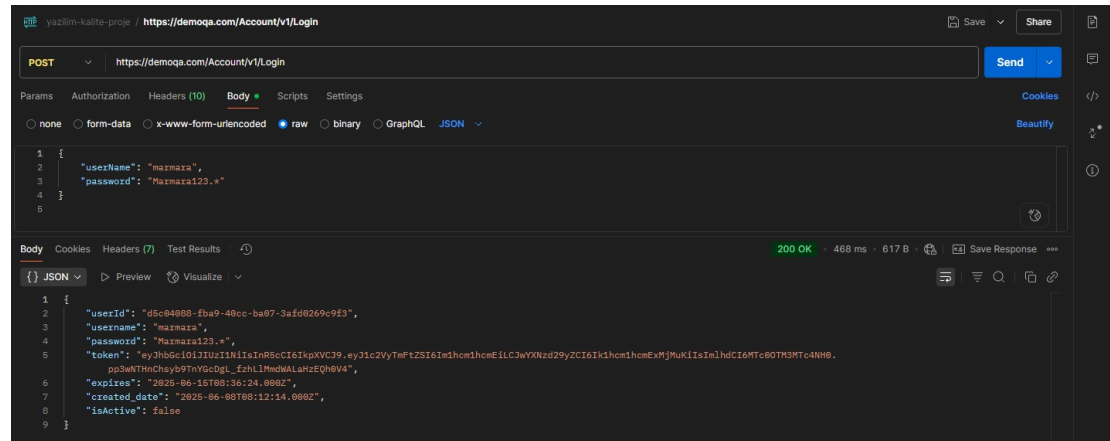


Kullanıcının kayıtları arasında bulunmayan isbn numaralı kitap silinmek istendiğinde gerçekleşen senaryo (400 Bad Request):

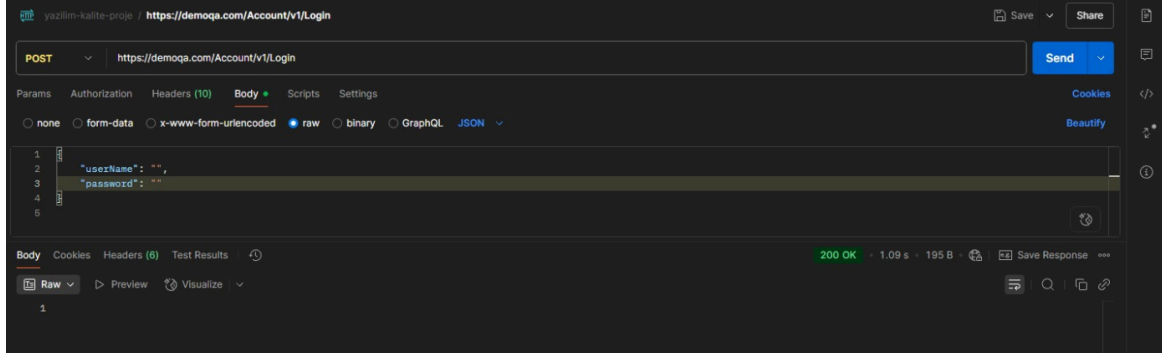


POST /Account/v1/Login = Kullanıcı giriş işlemini gerçekleştirir. Kullanıcı adı ve şifre doğrulaması yaparak başarılı giriş durumunda token veya oturum bilgisi sağlar.

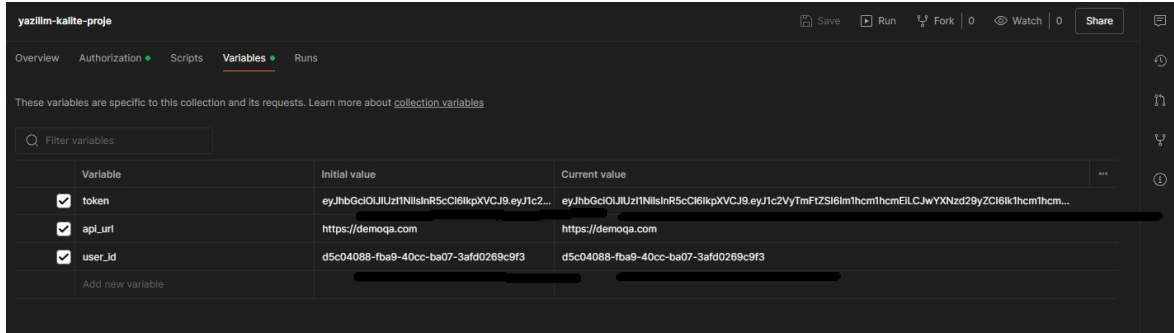
Kullanıcı tüm bilgilerini doğru girdiğinde oluşan senaryo (200 OK):



Kullanıcı bilgilerini yanlış girdiğinde veya girmediğinde farklı sonuçlar oluşması gerekirken status 200 OK olarak döndürülmektedir. Bu durum hatalı olup eğer hiçbir veri girilmediyse 400 Bad Request döndürülmesi gerekirken hatalı bir giriş yapmaya çalışıldığında 401 Unauthorized status kodu ile dönüş yapılmalıdır :



Aynı değerleri (örneğin API URL, token, kullanıcı ID gibi) birçok yerde tekrar tekrar yazmak yerine, değişken olarak tanımlayıp kolayca kullanabilirsiniz. Böylece sadece değişkenin değerini güncelleyerek tüm testlerde değişikliği hızlıca yapabilirsiniz.



3.2. UI Test Sonuçları (Selenium ile)


Kullanıcı arayüzüne yönelik testler, gerçek kullanıcı etkileşimlerinin otomatikleştirilerek simüle edilmesi yoluyla yürütülmüştür. Selenium WebDriver kullanılarak gerçekleştirilen bu testlerde, farklı senaryolar denenmiş ve uygulamanın kullanıcı deneyimi açısından ne derece tutarlı davrandığı gözlemlenmiştir.

Gerçekleştirilen UI test senaryoları şunlardır:

Kayıt Olma (Signup): Kullanıcı bilgileri form aracılığıyla girilmiş, CAPTCHA alanı manuel olarak geçilmiş ve başarı mesajı kontrol edilmiştir.

Register

Register to Book Store

First Name :	<input type="text" value="Selin"/>
Last Name :	<input type="text" value="Yılmaz"/>
UserName :	<input type="text" value="Selin123"/>
Password :	<input type="password" value="*****"/>
<input type="checkbox"/> I'm not a robot  reCAPTCHA Privacy - Terms	

Register

Back to Login

User exists!

Giriş Yapma (Login): Kayıtlı kullanıcı bilgileri ile giriş yapılmış, kullanıcı adı ekran üzerinde doğrulanmıştır.

Login

Welcome,

Login in Book Store

UserName :	<input type="text" value="Selin123"/>
Password :	<input type="password" value="*****"/>

Login

New User

Invalid username or password!

Kitap Silme (Delete Book): Mevcut kitaplardan biri silinmiş, işlem sonrası değişiklik doğrulanmıştır.

```

options = Options()
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)

driver.get("https://demoqa.com/login")

driver.find_element(By.ID, "userName").send_keys("marmarauni")
driver.find_element(By.ID, "password").send_keys("Mar123**")
driver.find_element(By.ID, "login").click()

time.sleep(3)

try:
    delete_button = driver.find_element(By.ID, "delete-record-undefined")
    delete_button.click()

    time.sleep(1)

    ok_button = driver.find_element(By.ID, "closeSmallModal-ok")
    ok_button.click()
except Exception as e:
    print("Silme işlemi sırasında hata:", e)

time.sleep(2)
driver.save_screenshot("after_delete.png")

driver.quit()

```

Tüm Kitapları Silme (Delete All Books): Kullanıcının kütüphanesindeki tüm kitaplar silinmiş ve bu işlemin sonucunda kitap listesinin boş olduğu görülmüştür.

```

8 options = Options()
9 driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()), options=options)
10
11 driver.get("https://demoqa.com/login")
12 driver.find_element(By.ID, "userName").send_keys("marmarauni")
13 driver.find_element(By.ID, "password").send_keys("Mar123**")
14 driver.find_element(By.ID, "login").click()
15
16 time.sleep(5)
17
18 try:
19     delete_all_button = driver.find_element(By.XPATH, "//*[@text()='Delete All Books']")
20     delete_all_button.click()
21
22     time.sleep(3)
23
24     ok_button = driver.find_element(By.ID, "closeSmallModal-ok")
25     ok_button.click()
26
27 except Exception as e:
28     print("Hata oluştu:", e)
29
30 time.sleep(2)
31 driver.save_screenshot("after_delete_all_books.png")
32 driver.quit()

```

Form Doldurma: Demo form sayfası üzerinde çeşitli alanlar doldurulmuş ve başarılı bir şekilde gönderildiği ekran görüntüleriyle doğrulanmıştır.

Thanks for submitting the form

Label	Values
Student Name	Selin Yıldız
Student Email	selin@example.com
Gender	Female
Mobile	5551234567
Date of Birth	08 June,2025
Subjects	Maths
Hobbies	Sports, Music
Picture	
Address	Istanbul, Turkey
State and City	NCR Delhi

Close

İşlemlerin beklenen sürede ve doğru biçimde gerçekleştiği gözlemlenmiş, herhangi bir sistemsel gecikme ya da arayüz hatası ile karşılaşılmamıştır.