Teknoloji Fakültesi

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

YAZILIM KALİTE VE GÜVENCE TEMELLERİ FİNAL PROJE RAPORU

PROJE KONUSU: Spotify API ve Kullanıcı Arayüzü (UI) Testlerinin Otomasyonu: Postman ve Selenium Kullanarak Yazılım Kalite Güvencesi Uygulaması

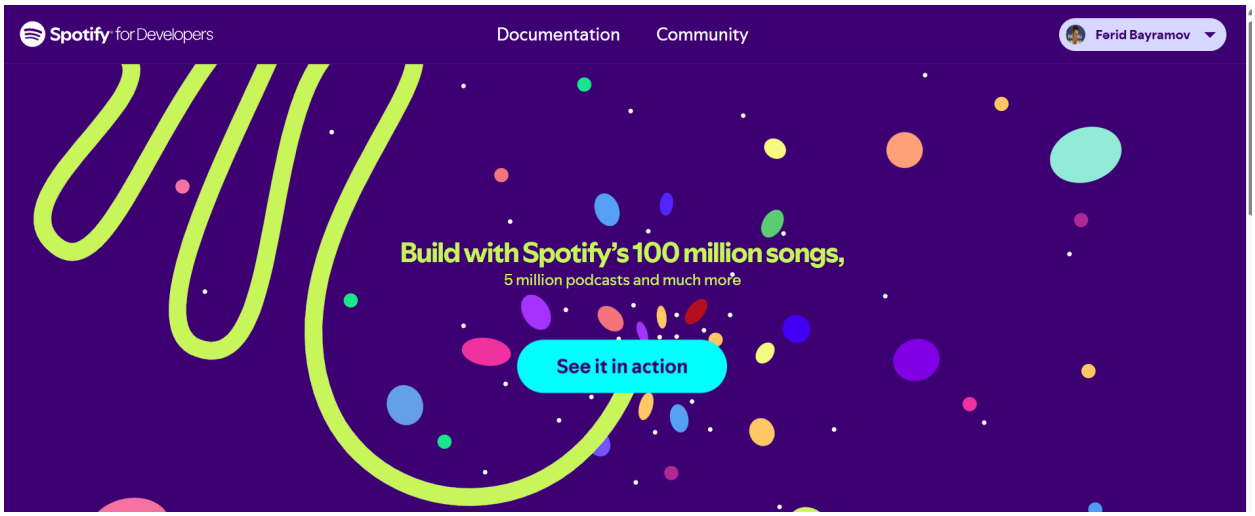
ÖĞRENCİLER:

Farid Bayramov	170421993
Emir Bince	170421923
Davutcan Kösemen	170421030

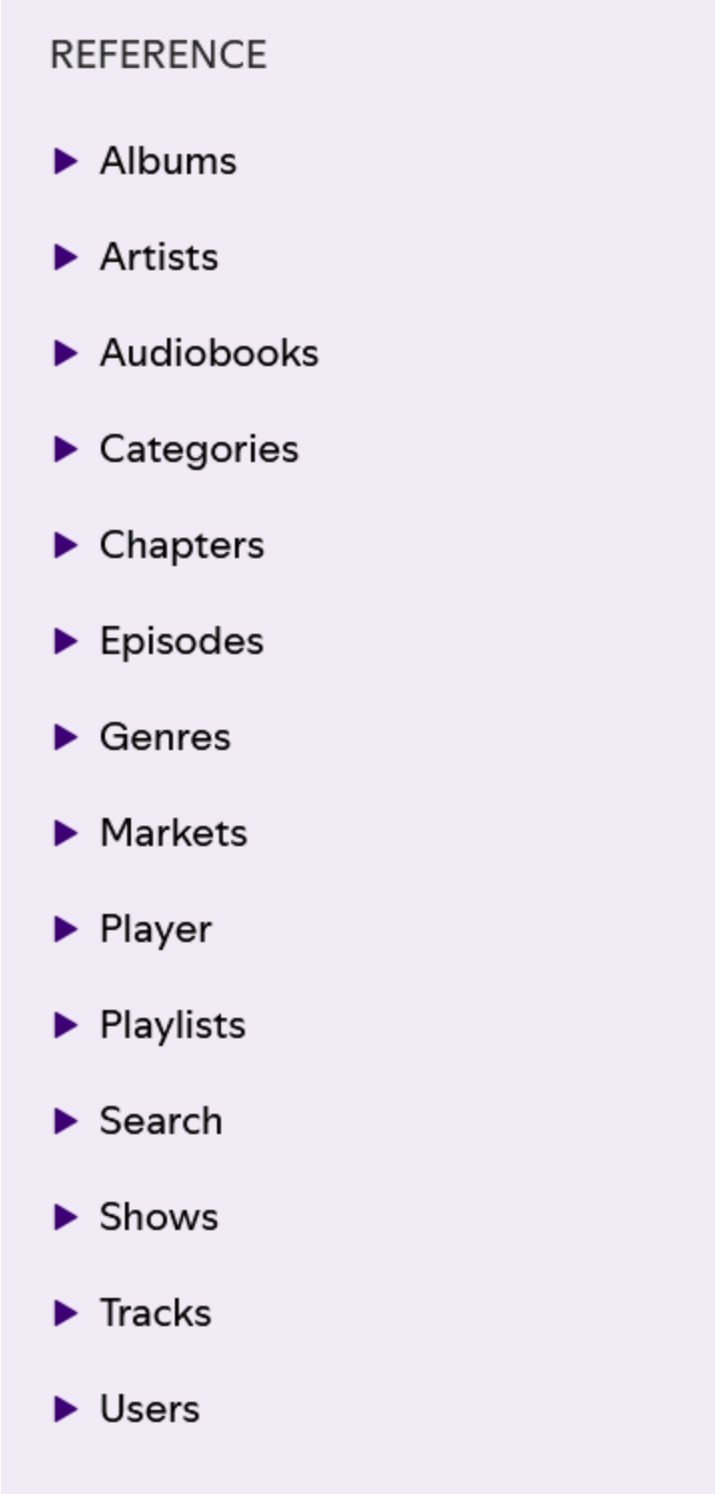
POSTMAN API

1. API Analizi ve Spotify Developer Hesabı Oluşturulması

Spotify API hizmetlerinden faydalanabilmek için ilk adım olarak https://developer.spotify.com adresi üzerinden bir Spotify Developer hesabı oluşturulmuştur. Bu hesap üzerinden uygulamaların yönetimi, client ID ve secret erişimi, redirect URI ayarları ve kullanım istatistikleri takip edilmektedir.



Spotify WEB API için developper hesabı oluşturulduktan sonra https://developer.spotify.com/documentation/web-api linkinden API'ın endpointleri ve bu endpointlerin aldığı parametreler, hata durumları, response ve kapsamları incelenmektedir.



Görsel 1 : Spotify API Endpointleri

Test Senaryolarımızın oluşturulması için öncelikle bir kullanıcı girişi gerekmektedir. Kullanıcı girişi yapıldıktan sonra o hesaba bağlı bilgileri

çekmek, Spotify'ın arama gibi özellikleri ve daha birçok endpoint okunduktan sonra 5 adet endpoint seçilmiştir. Bunlar aşağıdaki gibidir.

- Users
- Search
- Playlists
- Tracks
- Albums

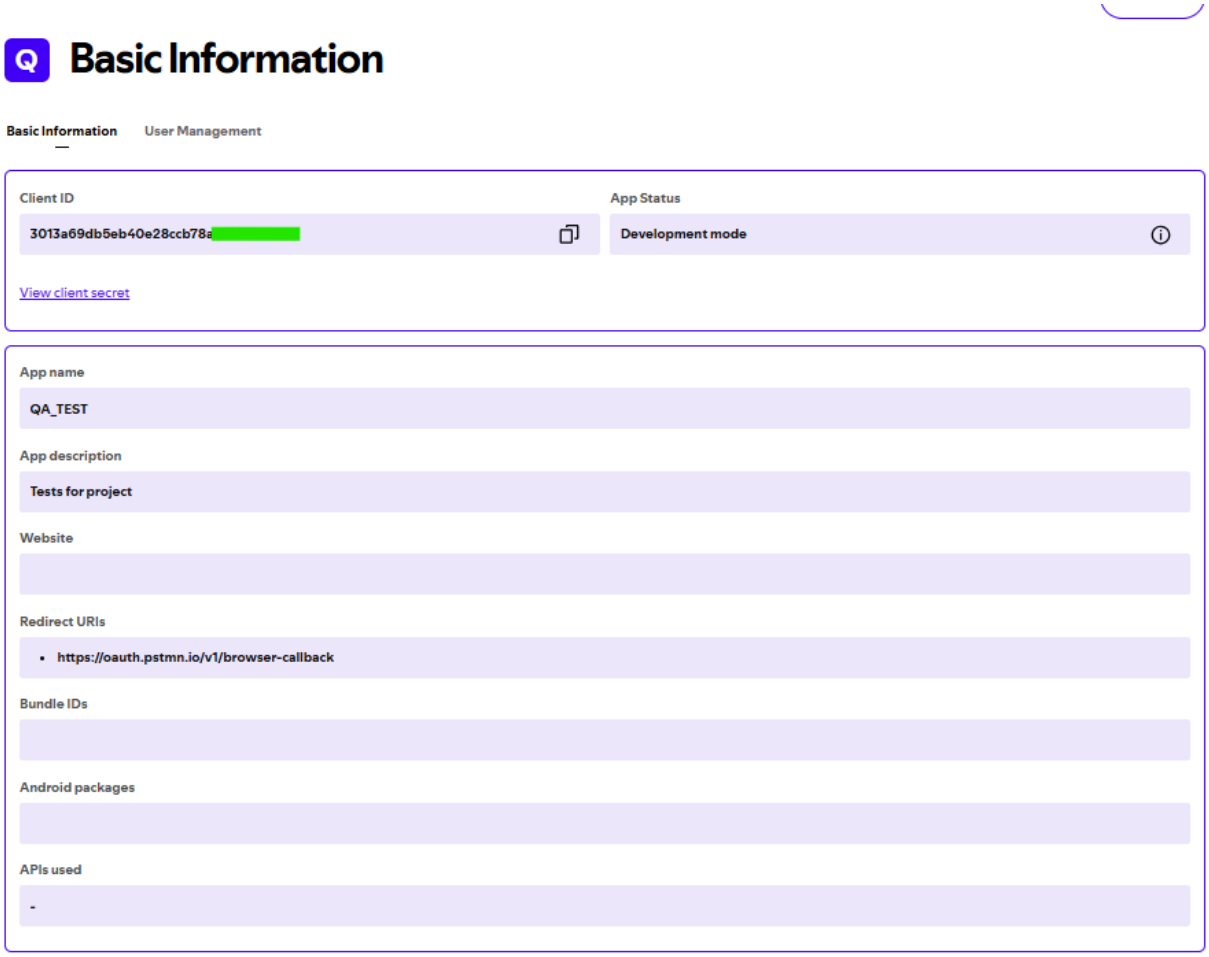
Bu endpointlerin hata durumları, otomatik test senaryoları, exception yönetimi ve responseların raporlanması planlanmaktadır.

2. Uygulama (App) Oluşturulması

Spotify Developer Dashboard üzerinden yeni bir uygulama (app) oluşturulmuştur. Bu işlem sırasında uygulamaya bir isim, açıklama ve geliştirici iletişim bilgileri girilmiştir.

Uygulama Bilgileri:

App Name: MySpotifyIntegration
Client ID: (otomatik oluşturuldu)
Client Secret: (otomatik oluşturuldu)
Redirect URI: *https://oauth.pstmn.io/v1/browser-callback*
Bu URI, OAuth akışında kullanıcı yetkilendirme işlemi sonrası token'ın döndüğü yerdir. Bu URI ayrıca **Spotify Dashboard**'a manuel olarak eklenmiştir.

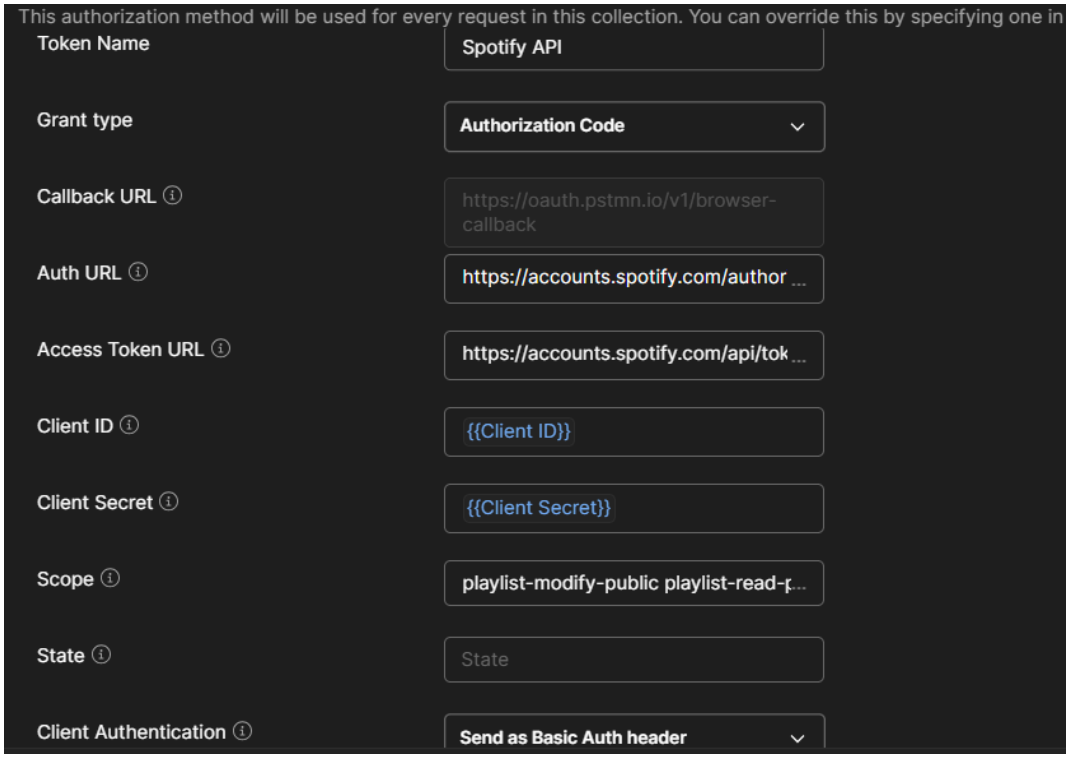


3. OAuth 2.0 ile Yetkilendirme Süreci

Spotify API, OAuth 2.0 protokolü ile çalışmaktadır. Bu protokol ile kullanıcıya ait verilere erişebilmek için önce **Access Token** alınması gereklidir. Bu amaçla aşağıdaki adımlar izlenmiştir:

3.1. Client Credentials (Token Alma İşlemi)

1. Yetkilendirme süreci Postman aracı üzerinden manuel olarak gerçekleştirilmiştir. Postman üzerinden **OAuth 2.0 Authorization** yöntemi kullanılarak aşağıdaki bilgiler girilmiştir:



2. **Scope**'lar ile kullanıcının profil bilgilerine, e-posta adresine, çalma listelerine, sık dinlenen sanatçılara ve kütüphanesine erişim izinleri sağlanmıştır.

3. Ardından **"Get New Access Token"** butonuna tıklanarak kullanıcıdan izin alınmış ve Spotify tarafından access token sağlanmıştır.

4. Gelen Access Token, **"Use Token"** butonuna tıklanarak API isteklerinde kullanılmak üzere seçilmiştir.

Bu işlem sonucunda erişim token'ı elde edilmiş ve **Spotify API** isteklerinde kullanılmıştır. Token alma işlemi tamamen Postman arayüzü üzerinden yürütülmüş, HTTP POST isteği manuel olarak yapılmamıştır.

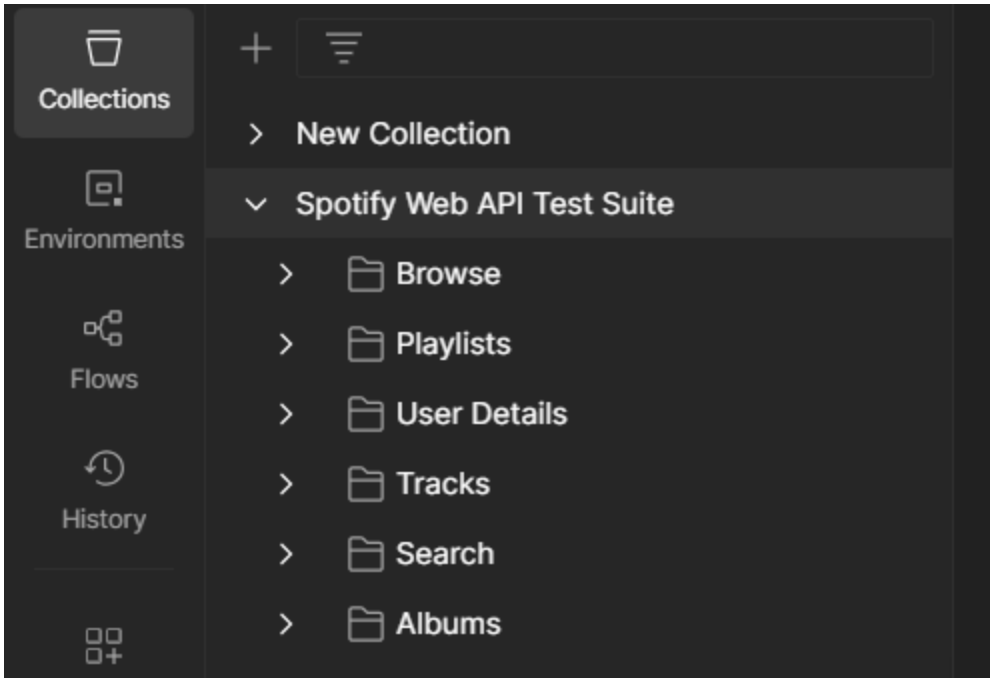
4. Postman'de Spotify Ortamı (Environment) ve Koleksiyon Yapılandırması

4.1. Spotify Environment (Çevre) Oluşturulması

Spotify API ile yapılan işlemlerin daha sistematik ve dinamik hale getirilmesi amacıyla Postman üzerinde özel bir **Environment (Çevre)** tanımlanmıştır. Bu environment içerisinde sıkça kullanılan sabit değerler **değişken (variable)** olarak saklanmış ve test isteklerinde doğrudan bu değişkenler çağrılmıştır.

4.2. Koleksiyon (Collection) ve Dizin (Folder) Yapısı

Postman üzerinde, testlerin düzenli bir yapıda ilerlemesi için bir **koleksiyon (collection)** oluşturulmuştur. Bu koleksiyon içerisinde kullanım senaryolarına göre klasörler (folders) organize edilmiştir.



5. API Test Süreci (Postman ile)

Access Token alındıktan sonra, seçilen Spotify API endpoint'leri Postman üzerinden test edilmiştir. Bunlar için collectionda açılan requestler kullanılıp cevap durumları, otomasyon testlerini geçip geçmediği ve dokümantasyonda belirtilen cevapları alıp almadığı test edilmiştir. Aşağıdaki durum kodları beklenmektedir :

200 - Başarılı durum sonucundaki durum kodudur.

401 - Kötü veya Süresi geçmiş token sonucunda dönen durum kodudur.

403 - Yanlış doğrulama ve token eşleşme sonucunda dönen durum kodudur.

429 - API istek aşımı veya çalışmama durumunda dönen durum kodudur.

1. Albums - Yeni Çıkan Albüm Kontrolü

Request

▼ GET /browse/new-releases

limit integer

The maximum number of items to return. Default: 20. Minimum: 1. Maximum: 50.

Default: `limit=20`
Range: `0 - 50`
Example: `limit=10`

offset integer

The index of the first item to return. Default: 0 (the first item). Use with limit to get the next set of items.

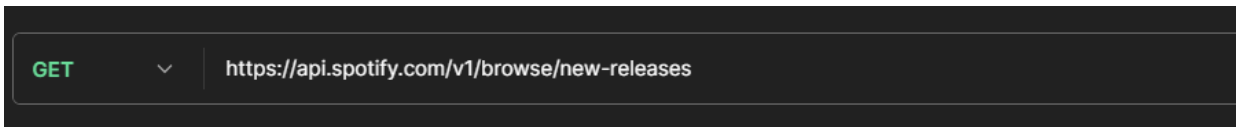
Default: `offset=0`
Example: `offset=5`

Response 200 401 403 429

A paged set of albums

▶ **albums** object Required

Bu istek, Spotify'da yeni yayınlanmış albüm türündeki içerikleri getirir. Genellikle ülkeye ve isteğe göre sayfalama (limit/offset) desteğiyle sonuç döner. **GET** isteği kullanıldı. 200 status code ve beklendiği çıktı alındı.



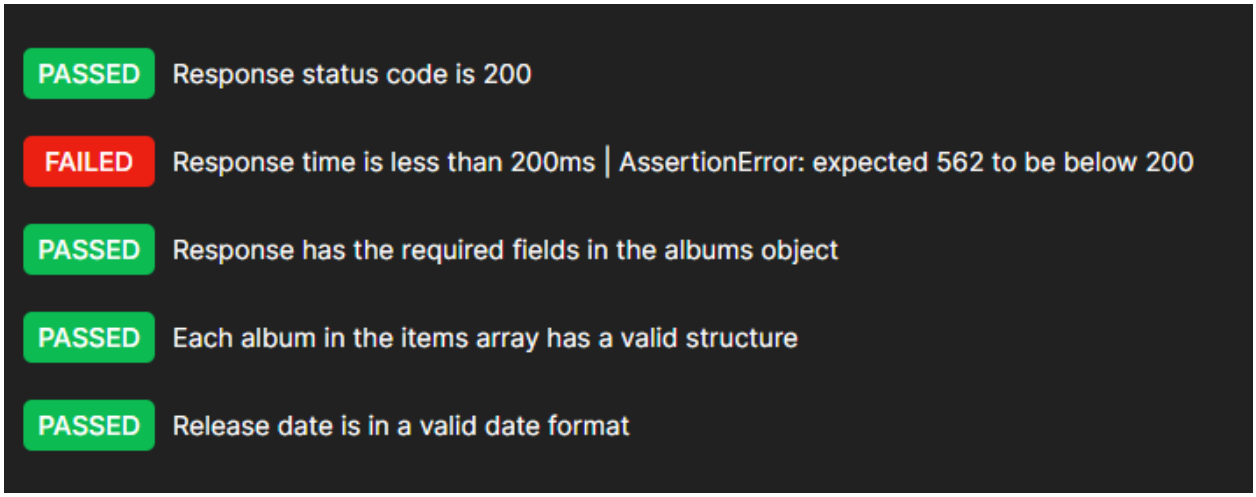
Otomasyon ve Hatalı Durum Testi

Burada örnek olması için Otomasyon testi olarak dönen tarih formatı, içerik sırası ve ülke gibi bilgilerin doğru yapıda olup olmadığı post-reqeust test scripti ile kontrol edilmiştir.

```
pm.test("Release date is in a valid date format", function () {
    const responseData = pm.response.json();
    pm.expect(responseData).to.be.an('object');

    responseData.albums.items.forEach(item => {
        pm.expect(item.release_date).to.match(/^\d{4}-\d{2}-\d{2}$/, "Release date should
    });
});
```

Test gerçekleştirildiğinde ise 5 testten 4 ünü geçtiği, sadece kullanım yoğunluğu gibi durumlara göre değişmekte olan response time da kaldığı görülmektedir. İçerik formatları ve durum kodu gibi kontrol testlerinden başarılı şekilde geçtiği görülmektedir.



Hatalı kullanım durumunda API kendi içerisinde boş kalan bilgiyi default ayarlarından aldığı için bu durumda endpointin hatalı kullanımı yalnızca

yanlış-geçersiz token durumları kontrol edilmektedir. Yukarıda belirtildiği gibi geçersiz token durumlarında 401 kodu döndüğü görülmektedir.

```
{
    "error": {
        "status": 401,
        "message": "No token provided"
    }
}
```

2. Playlist - Playlist'e belirli bir müzik ekleme

Request

▶ **POST** `/playlists/{playlist_id}/tracks`

▼ **Body** application/json

supports free form additional properties

uris array of strings

A JSON array of the Spotify URIs to add. For example: `{"uris": ["spotify:track:4iV5W9uYEdYUVa79Axb7Rh","spotify:track:1301WleyT98MSxVHPZCA6M", "spotify:episode:512ojhOuo1ktJprKbVcKyQ"]}`
A maximum of 100 items can be added in one request. ***Note**: if the `uris` parameter is present in the query string, any URIs listed here in the body will be ignored.*

position integer

The position to insert the items, a zero-based index. For example, to insert the items in the first position: `position=0`; to insert the items in the third position: `position=2`. If omitted, the items will be appended to the playlist. Items are added in the order they appear in the uris array. For example: `{"uris": ["spotify:track:4iV5W9uYEdYUVa79Axb7Rh","spotify:track:1301WleyT98MSxVHPZCA6M"], "position": 3}`

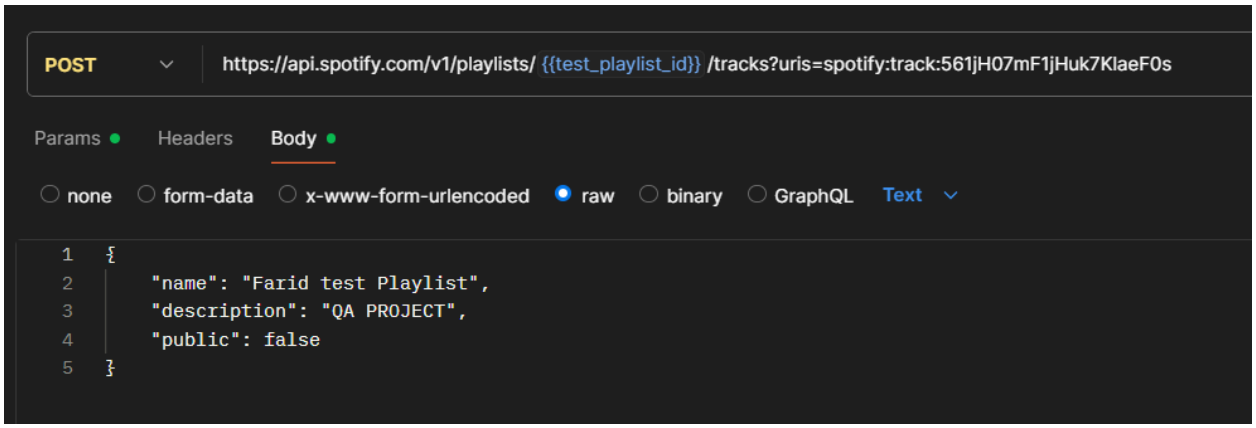
Response 201 401 403 429

A snapshot ID for the playlist

snapshot_id string

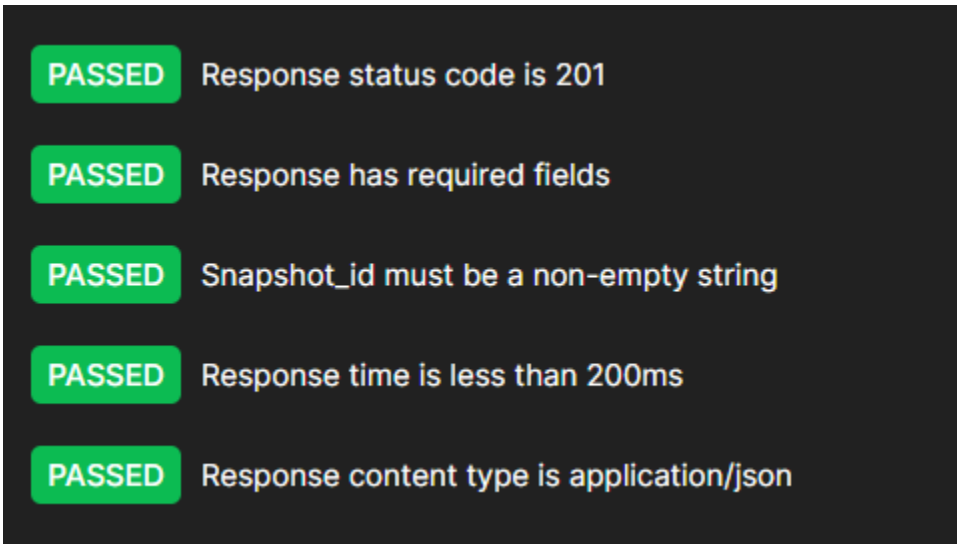
Example: `"abc"`

Request body olarak albümün adını , tanımını, public durumu **POST** metodu ile gönderilir. Bu sayede kendi playlistimize müzik eklenmiş olur. Geriye ise playlistin snapshot idsi döndürülür.

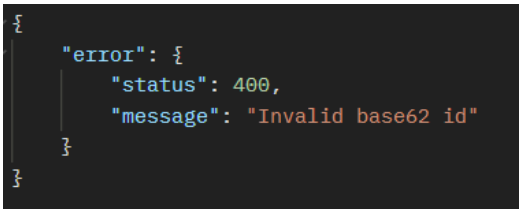


Otomasyon ve Hatalı Durum Testi

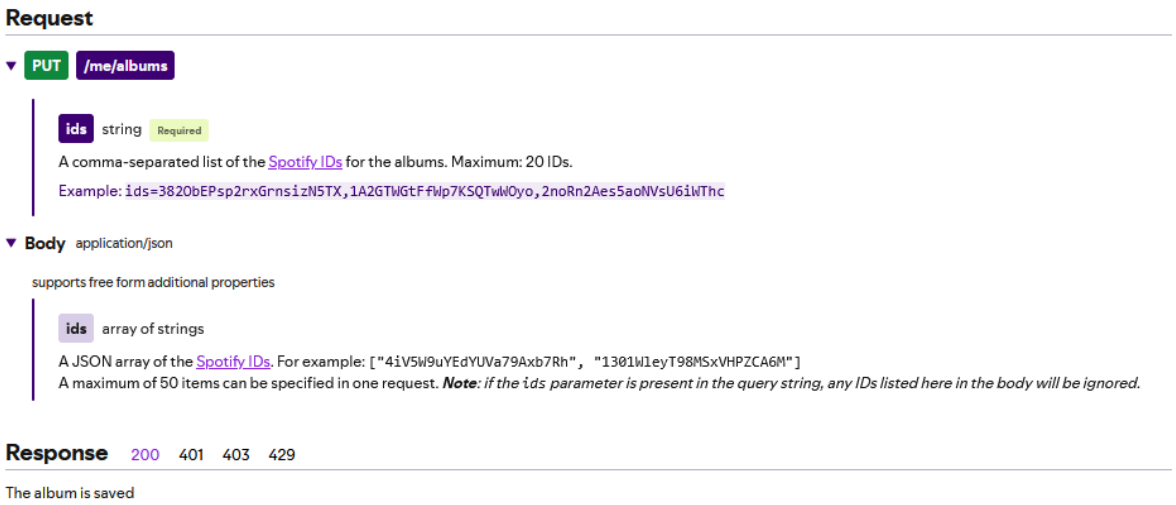
Burada dönen snapshot_id nin integer formatında olması, response time ve api koşullarına uygunluğu test edilmektedir. Otomasyon testi sonucunda istenen alanların geldiği ve başarıyla çalıştığı görülmektedir.



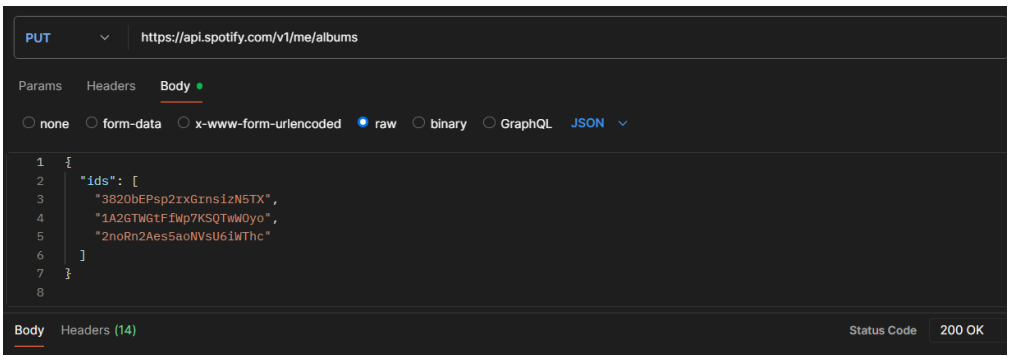
Hatalı durumda ise geçersiz veya hiç şarkı bilgisi göndermeme hali örnek alınarak test edilmiştir bu durumda ise dönen cevaplar aşağıdaki gibi olmaktadır.



3. Albums - Mevcut kullanıcı için albümleri kaydetme



Albüm kayıt endpointinde ise Spotifyda albümleri kaydetme özelliği kullanılır. Seçilen albümlerin idsi request içerisinde **PUT** metodu ile gönderilir ve sonuçları kontrol edilir.



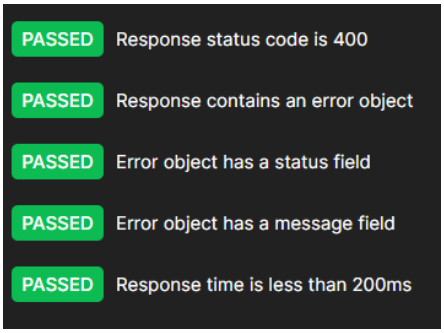
Otomasyon ve Hatalı Durum Testi

Otomasyon için gönderilen id ler sonucunda dönen objede bir hata objesi olup olmadığı, mesaj içerip içermediği gibi testler uygulanmaktadır.

```
pm.test("Error object has a message field", function () {
    const responseData = pm.response.json();

    pm.expect(responseData).to.be.an('object');
    pm.expect(responseData.error).to.exist.and.to.be.an('object')
    pm.expect(responseData.error).to.have.property('message');
});
```

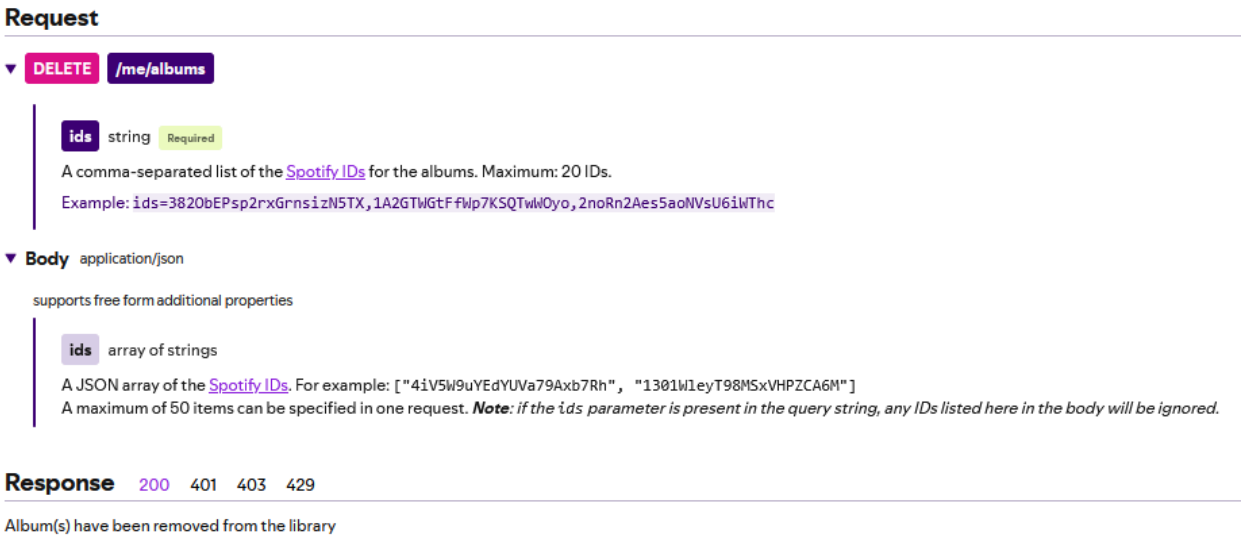
Bu ve durum kodu, response time gibi değerler kontrol edilerek testlerin doğru çalışıp çalışmadığı onaylanmıştır.



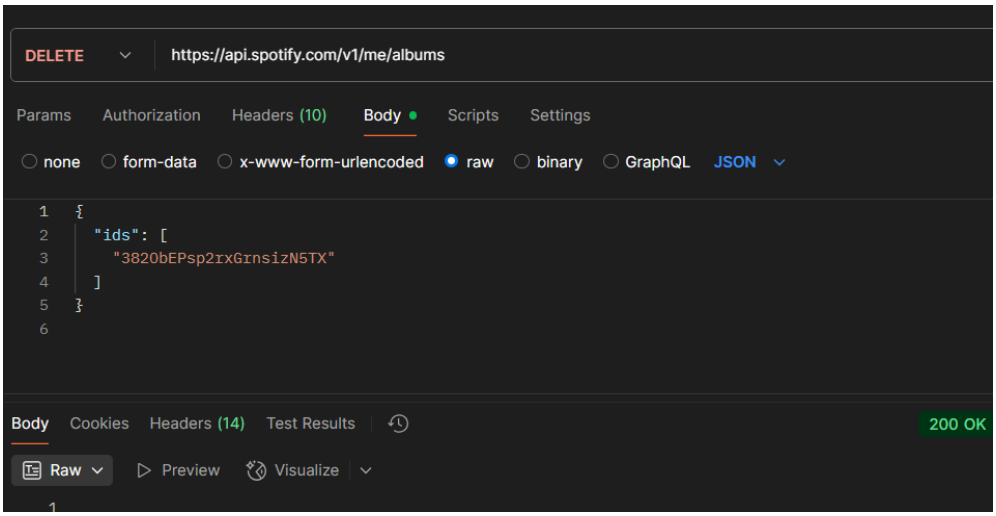
Hatalı durum ise body içerisinde gönderilen ids bilgisinin olmayışı/hatalı girişi test olarak alınmaktadır. Bu durumda ise geriye 400 durum kodlu aşağıdaki hata mesajı dönmektedir.

```
{
    "error": {
        "status": 400,
        "message": "Missing required field: ids"
    }
}
```

4. Albums - Albüm silme

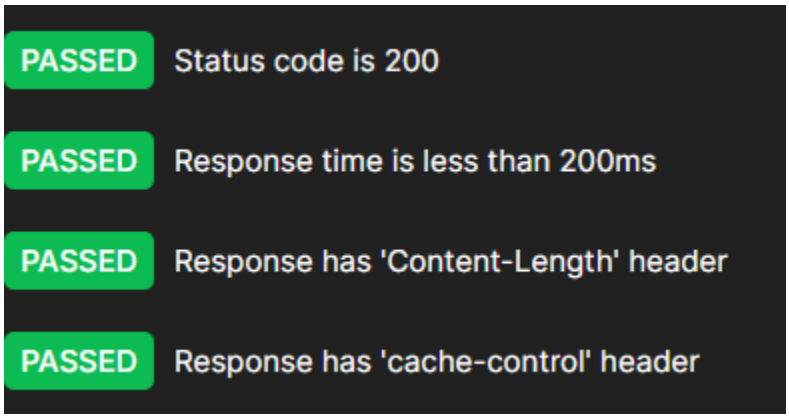


Albümleri silmek için kullanılan endpointte ise request içerisinde silinmesi istenen albüm id si **DELETE** metodu ile gönderilir.

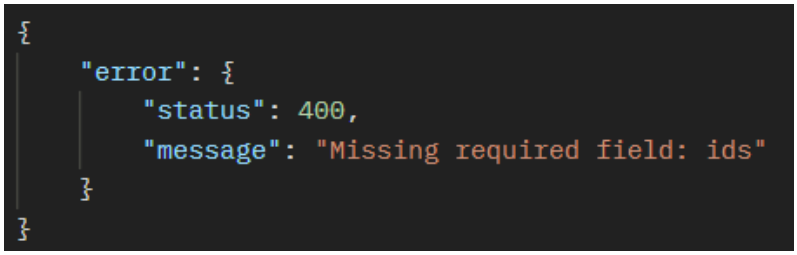


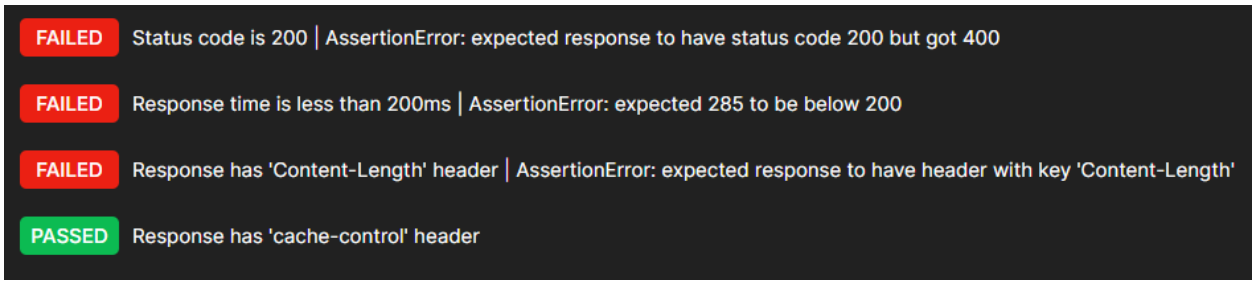
Otomasyon ve Hatalı Durum Testi

Otomatik test scripti için kısaslar bir önceki endpointte olduğu gibi hata dönüp dönmediği, bir mesaj içerip içermediği ve response time gibi metriklerden oluşmaktadır. Aşağıda ise bu testlerden başarılı şekilde geçtiği görülmektedir.



Hata senaryoları ise ids bilgilerinin gönderilmeyişi test edilmekte ve aşağıdaki cevap alınmaktadır.





Burada görüldüğü gibi gerekli bilgi bodyde gönderilmediği durumlarda response yukarıdaki gibi hata bilgisi ve 400 durum kodu döndürmektedir.

5. Tracks - Katalog Bilgileri Görüntüleme

Request

▶ GET /tracks/{id}

Response 200 401 403 429

A track

▶ **album** object

The album on which the track appears. The album object includes a link in `href` to full information about the album.

▶ **artists** array of SimplifiedArtistObject

The artists who performed the track. Each artist object includes a link in `href` to more detailed information about the artist.

available_markets array of strings

A list of the countries in which the track can be played, identified by their ISO 3166-1 alpha-2 code.

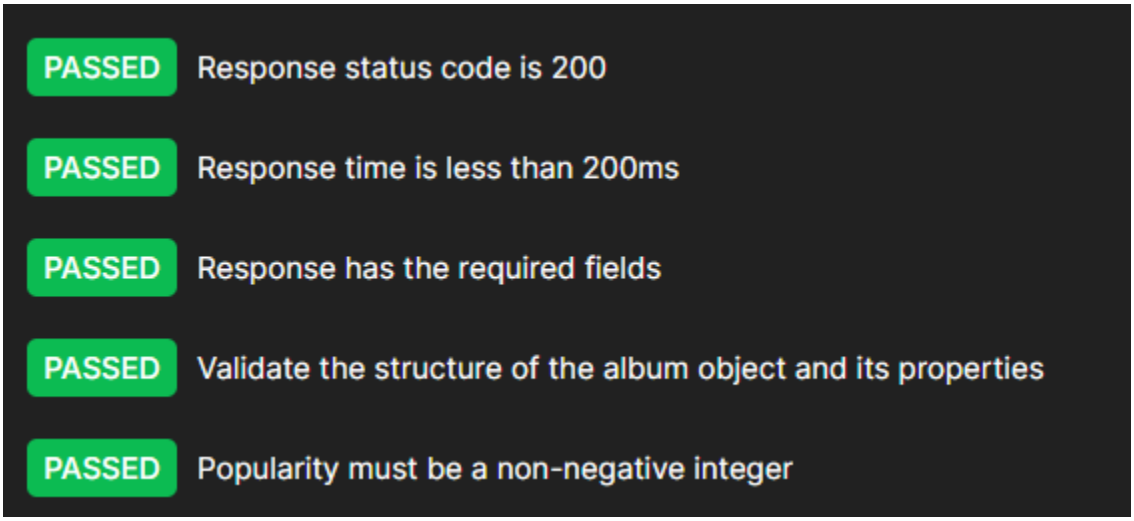
Dokümantasyonda görüldüğü üzere id ve market bilgisi request içerisinde **GET** isteği gönderildiğinde album, artist ve available_markets vs. bilgileri dönmektedir. Bu linke **GET** isteği atılır ve gelen sonuç kontrol edilir.


```
{
    "album": {
        "album_type": "album",
        "artists": [
            {
                "external_urls": {
                    "spotify": "https://open.spotify.com/artist/0C0XlULifJtAgn6ZNCW2eu"
                },
                "href": "https://api.spotify.com/v1/artists/0C0XlULifJtAgn6ZNCW2eu",
                "id": "0C0XlULifJtAgn6ZNCW2eu",
                "name": "The Killers",
                "type": "artist",
                "uri": "spotify:artist:0C0XlULifJtAgn6ZNCW2eu"
            }
        ],
        "available_markets": [
            "AR",
            "AU",
            "AT",
```


Beklenen parametreleri karşıladığı için bu API'nın uygun koşullarda çalıştığı görülmektedir.

Otomasyon ve Hatalı Durum Testi

Bu endpoint için katalog bilgileri görüntülemede body içerisinde gönderilen bilgilerin, response ile dönen içeriklerin format uygunluğu, içerik doğruluğu ve diğer performans metriklerin durumuna göre test scripti yazılmaktadır. İstenen id ve market bilgisi API içerisinde boş geldiği durumlarda API kendi içeriside kullanıcı bilgilerine erişip gerekli alanları doldurduğu için burada kontrol edilen hata durumları yanlış track id girilmesidir.



Hatalı track id durumunda ise aşağıdaki mesaj döndüğü görülmektedir

SELENIUM UI TESTLERİ

Kullanıcı etkileşimlerini bir makinede simüle etmek için seçilen web sitesi, aşağıdaki bağlantıdan erişilebilen Spotify'ın arama sayfasıdır:
"https://open.spotify.com/search"
Yapılan testler arasında web sayfasını açmak, belirli sanatçıları aramak, en iyi parça adlarını çıkarmak ve sanatçı sayfalarının ekran görüntüsünü kaydetmek yer alır. Ayrıntılı açıklama aşağıda listelenecektir.

1. **Spotify Arama Sayfasını Aç**
   Projede, Selenium testlerini yürütmek için Chrome webdriver kullanıldı. Bu nedenle, Spotify arama sayfasına erişmek için webdriver'ın get fonksiyonu kullanıldı.

```
driver = webdriver.Chrome(options=options)
wait = WebDriverWait(driver, 10)

try:
    # 1. Open Spotify search page
    driver.get("https://open.spotify.com/search")
    print("Opened Spotify search page.")
    time.sleep(5)
```

   Beş saniyelik bir uyku zamanlayıcısı sunucuyu bunaltmaktan kaçınmak için ayarlanır ve aynı zamanda sayfadaki her şeyin düzgün bir şekilde yüklenmesi için zaman verildiğinden emin olmak için kullanılır. Bu adım, betiğin amaçlandığı gibi çalıştığından emin olmak için çok önemlidir.

2. **Çerezleri Kabul Et**
   Spotify arama web sitesi ilk açıldığında, ön uç kullanıcı arayüzü kullanıcıdan çerezleri kabul etmesini veya reddetmesini ister. Bu açılır pencere web sitesindeki gerçek içeriği gizleyebilir, bu nedenle açılır

penceredeki "çerezleri kabul et" düğmesine tıklayarak bundan kurtulmak önemlidir.

```
# 2. Accept cookies if the button is present
try:
    cookie_button = wait.until(EC.element_to_be_clickable((By.XPATH, "/
    cookie_button.click()
    print("Accepted cookies.")
except:
    print("No cookie consent button found or already accepted.")

time.sleep(5)
```

Yukarıdaki görüntü, açılır pencere sunulduğunda çerezleri kabul etmek için kullanılan kod parçacığını gösterir. Kodun, çerez onay düğmesinin algılanmaması durumunda bir geri dönüş sistemine sahip olduğunu unutmayın. Bu durumda, işlevlerine normal şekilde devam edecektir.

3. **Bir Sanatçı Ara**

Motor, Spotify arama web sitesinde giriş kutusunu bulacak ve betiği çalıştırmadan önce kullanıcı tarafından belirlenen bir sanatçının adını girecektir. Bu eylem, önce sanatçının adını gönderip ardından enter'a basarak "send_keys" işlevini kullanarak gerçekleştirilir. Bu eylem, elbette, Spotify'daki sanatçıları arar, ardından web sitesinin düzgün yüklendiğinden emin olmak için 5 saniyelik bir uyku zamanlayıcısı daha vardır.

```
# 3. Search for an artist
search_artist = 'Michael Jackson'
search_input = wait.until(EC.presence_of_element_located((By.TAG_NAME,
search_input.send_keys(search_artist)
search_input.send_keys(Keys.RETURN)
print(f"Searched for {search_artist}.")
time.sleep(5)
```

“search_artist” değişkeni kullanıcı tarafından istediği sanatçıyı içerecek şekilde değiştirilebilir. Yukarıdaki görüntüde, örnek olarak "Michael Jackson" kullanılmıştır.

Arama sorgusu düzgün bir şekilde yüklendikten sonra, sanatçı adları hemen görünür olmadığından ve kullanıcının bunları görüntülemek için aşağı kaydırmasını gerektirdiğinden, motor sanatçı bölümünü yüklemek için biraz aşağı kaydırır. Bu eylem aşağıdaki komutla otomatikleştirilir:

```
# 4. Scroll down a bit to load the artist section
driver.execute_script("window.scrollTo(0, 600);")
time.sleep(3)
```

Aşağı kaydırma, sanatçı bölümünün yüklenmeye başlamasını garanti eder ve uyku zamanlayıcısı, düzgün bir şekilde yüklenmeden önce hiçbir komutun çalıştırılmamasını sağlar.

4. **Sanatçı Kartına tıklayın**

Belirli bir sanatçıyı aradıktan ve sanatçı bölümünü yüklemek için aşağı kaydırdıktan sonra, motor tıklanabilir öğedeki metni aranan sanatçı adıyla eşleştirerek ilk sanatçı kartına tıklayacaktır. Bu eylem sanatçının profil sayfasını yüklemeye başlayacaktır.

```
# 5. Click on the first artist card by partial match of href or text
artist_card = wait.until(EC.element_to_be_clickable((By.XPATH, f"//a[cor
artist_card.click()
print(f"Clicked on {search_artist}'s artist page.")
time.sleep(5)
```

Sanatçının sayfasını yükledikten sonra sanatçının parçalarına ulaşmak için sanatçının sayfasını aşağı kaydırmak için benzer bir yöntem kullanılacak ve sanatçının tüm sayfası yüklenecektir.

```
# 6. Scroll down on artist page
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
print("Scrolled to bottom of artist page.")
time.sleep(3)
```

Bu eylem, sanatçının parçalarının çıkarılmasının temelini oluşturur.

5. **En İyi Parçaları Çıkar**

Artık sanatçının profil sayfası doğru şekilde yüklendiğine göre, sanatçının ilk beş parça başlığı çıkarılabilir ve konsola yazdırılabilir. Ne yazık ki, Spotify'ın kullanıcı arayüzü, oturum açma biçiminde uygun yetkilendirme olmadan ses oynatmaya izin vermiyor. Oturum açma sayfasının kullandığı CAPTCHA nedeniyle, Selenium aracısı oturum açma görevini kendi başına tamamlayamaz. Bu sistemi atlatmak için yöntemler kullanılabilirdi ancak bu, projeyi olması gereken şeyden uzaklaştırırdı. Sonuç olarak, oturum açma işlevi yukarıda belirtilen nedenlerden dolayı kullanılmadı.

```
# 7. Extract top track names
print("Top tracks:")
track_titles = driver.find_elements(By.XPATH, "//div[@data-testid='track
for i, track in enumerate(track_titles[:5], start=1):
    print(f"{i}. {track.text}")
```

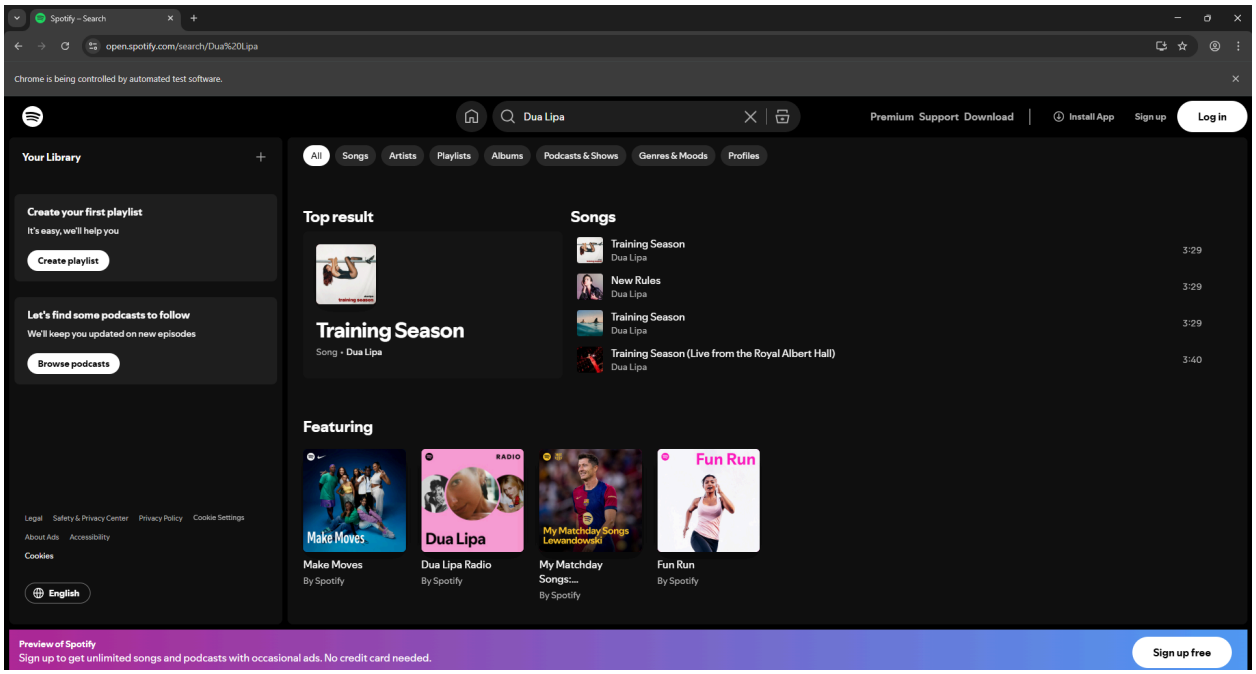
Yukarıdaki görsel, sanatçının profil sayfasından ilk beş parçayı çıkarmak için kullanılan kod bloğunu göstermektedir.

Son olarak, kullanıcı arayüzünden parça adlarını çıkardıktan sonra, sanatçının sayfasının ekran görüntüsü “saved_artist_pages” adlı bir klasöre kaydedilecektir. Bu eylem, Chrome webdriver nesnesinin “save_screenshot” işlevi kullanılarak gerçekleştirilecektir.

```
# 8. Take a screenshot
screenshot_path = f"saved_artist_pages/{search_artist}_artist_page.png".
driver.save_screenshot(screenshot_path)
print(f"Screenshot saved: {screenshot_path}")
```

DEMO

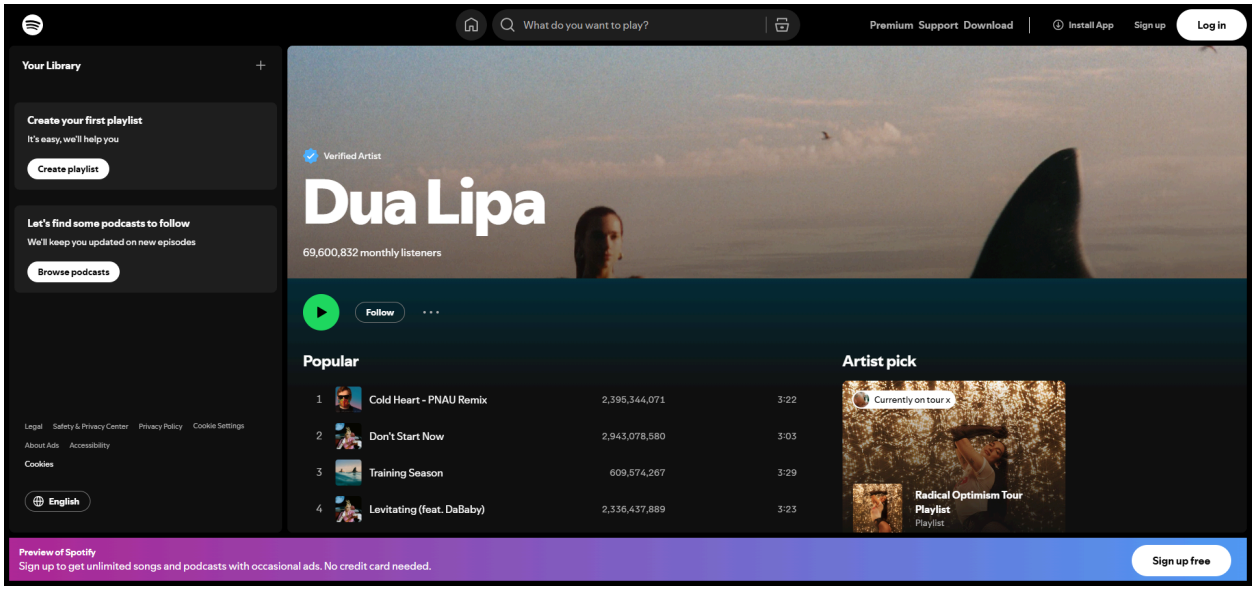
Komut dosyasının gösterimi, programın nasıl çalıştığını ve verileri nasıl çıkardığını görsel olarak açıklayacaktır. Bu örnek için, arama için sanatçı "Dua Lipa" kullanılacaktır.



Terminal çıktısı şu şekildedir:

```
Searched for Dua Lipa.
Created TensorFlow Lite XNNPACK delegate for CPU.
Attempting to use a delegate that only supports static-sized tensors with a graph that has dyna
mic-sized tensors (tensor#-1 is a dynamic-sized tensor).
Clicked on Dua Lipa's artist page.
Scrolled to bottom of artist page.
Top tracks:
1. Cold Heart - PNAU Remix
2. Don't Start Now
3. Training Season
4. Levitating (feat. DaBaby)
5. One Kiss (with Dua Lipa)
Screenshot saved: saved_artist_pages/dua_lipa_artist_page.png
```

Son olarak kaydedilen ekran görüntüsü aşağıda görüntülenebilir:



Programın koduna buradan ulaşabilirsiniz:https://github.com/FariddBayramov/QA_Project/tree/main