Marmara Üniversitesi Teknoloji Fakültesi

Yazılım Kalite Güvence Temelleri

Proje Raporu

Ayça Nilay Özoğul *170421012*

Erdem Saçan *170422826*

Hamza Gençay *170422822*

Merve Hasaslari *170421922*

Özet

Bu rapor, bir yazılım sistemine yönelik gerçekleştirilen API ve kullanıcı arayüzü (UI) testlerinin kapsamlı bir analizini sunmaktadır. Testler iki ana bölümde incelenmiştir: **API test senaryoları** ve **Selenium tabanlı UI test senaryoları**. API testleri, sistemin farklı senaryolara verdiği yanıtları değerlendirerek işlevselliğini, güvenilirliğini ve performansını ölçmeyi amaçlamaktadır. Bu kapsamda, geçerli ve geçersiz şehir adlarıyla yapılan istekler, dil parametrelerinin etkisi, hata kodlarının doğruluğu, yanıt süreleri, veri tiplerinin tutarlılığı, aşırı istek durumundaki sistem kararlılığı ve güvenlik açıklarının (örneğin, API anahtarı doğrulaması, XSS saldırılarına karşı dayanıklılık) olup olmadığı test edilmiştir.

Testlerin önemli bir bölümü, kullanıcının sisteme geçerli şehir adı girmesi durumunda dönen hava durumu verilerinin doğruluğunu ve tamlığını kontrol etmeye yöneliktir. Ayrıca, sistemin varsayılan dil olarak İngilizce döndürdüğü metinlerin, parametre kullanılarak Türkçe olarak elde edilip edilemeyeceği test edilmiştir. API'nin geçersiz şehir adı durumunda 404 Not Found döndürmesi, hatalı veya eksik API anahtarı kullanımı durumunda ise 401 Unauthorized hatası vermesi gibi güvenlik ve hata yönetimi senaryoları detaylı şekilde incelenmiştir. Sistem, veri türlerinin beklenen formatta olup olmadığının kontrol edilmesiyle de test edilmiştir (örneğin, sıcaklık verisinin float, açıklamanın string olması).

Öte yandan, **Selenium kullanılarak hazırlanan UI test senaryoları**, kullanıcı etkileşimlerine dayalı olarak sistemin arayüz tepkilerini ve işlevlerini doğrulamak amacıyla gerçekleştirilmiştir. Excel dosyasında yer alan bu senaryolarda; kullanıcı giriş ekranı, arama alanı, butonların doğru çalışıp çalışmadığı, sayfa yönlendirmeleri, elementlerin görünürlük durumları ve kullanıcı aksiyonlarına karşı sistemin verdiği tepkiler detaylı olarak test edilmiştir. Kullanıcı deneyimini artırmaya yönelik bu testlerde özellikle, UI elementlerinin doğru sırayla yüklenip yüklenmediği, dinamik içeriklerin sorunsuz çalışıp çalışmadığı ve hata mesajlarının kullanıcıya açık şekilde iletilip iletilmediği gibi unsurlar değerlendirilmiştir.

Yapılan tüm bu testler sonucunda, sistemin büyük ölçüde işlevsel ve güvenilir olduğu, ancak bazı senaryolarda (örneğin, aşırı istek yüklemesi altında gecikmeler veya eksik hata mesajları gibi) iyileştirmeye açık alanların bulunduğu görülmüştür. Elde edilen bulgular, sistemin daha kararlı ve kullanıcı dostu hale gelmesine katkı sağlayacak şekilde yorumlanmış ve öneriler geliştirilmiştir.

Abstract

This report presents a comprehensive analysis of API and user interface (UI) testing conducted on a software system. The tests are categorized into two main sections: API testing scenarios and Selenium-based UI test cases. The objective of the API tests is to evaluate the system's functionality, reliability, and performance under various conditions, including the use of valid and invalid city names, language parameter effects, response correctness, security validation through API key handling, data type consistency, stress testing, and defense against potential vulnerabilities like JavaScript injection (XSS).

A key focus of the API testing is verifying whether the system returns accurate and complete weather data when a valid city name is provided. Furthermore, the tests examine if weather descriptions can be localized (e.g., from English to Turkish) by modifying the API parameters. Error handling is rigorously tested, particularly ensuring the system returns a 404 Not Found error for invalid city names and a 401 Unauthorized error for requests lacking a valid API key. Data integrity is also assessed, with checks to confirm fields such as temperature and weather descriptions adhere to expected data types (e.g., float and string, respectively).

On the other hand, the Selenium-based UI tests aim to verify the front-end behavior of the application through user interaction simulations. The test scenarios listed in the accompanying Excel file cover critical UI elements such as login screens, search functionality, button behavior, page redirections, element visibility, and dynamic content loading. These tests focus on ensuring that UI components function correctly, are rendered in the right sequence, and provide appropriate feedback or error messages to the user when needed.

The results indicate that the system is largely stable and functional. However, several areas for improvement have been identified, especially under high-load conditions and in scenarios involving incomplete error responses. The insights derived from these tests offer valuable guidance for enhancing the system's robustness, security, and user experience.

1. **API Test Raporu – API-04 (Geçersiz Şehir Sorgusu)**

Senaryo No	Açıklama	Giriş Verisi	Beklenen Çıkış	Beklenen Sonuç
API-01	Geçerli şehir ile GET isteği	q=London	HTTP 200	JSON içinde 'name': 'London'
API-02	Birim dönüşümü testi	q=London&units=metric	HTTP 200	'main.temp' alanı °C cinsinden
API-03	Dil parametresi ile istek	q=London&lang=tr	HTTP 200	'description' Türkçe olmalı
API-04	Geçersiz şehir ismi	q=asdfgh	HTTP 404	'message': 'city not found'
API-05	API key eksik gönderimi	q=London	HTTP 401	'Invalid API key' mesajı
API-06	Yanıt veri tipi kontrolü	q=London	HTTP 200	'main.temp' float, 'weather' string
API-07	Boş şehir parametresi	q=	HTTP 400	'Nothing to geocode' mesajı
API-08	100 ardışık istek	q=London (loop 100x)	HTTP 200	Cevap süresi < 1s, hata yok
API-09	Eş zamanlı istekler	farklı şehirlerle paralel istek	HTTP 200	API kararlı yanıt vermeli
API-10	API brute force	20 geçersiz API key	HTTP 401	IP block ya da tekrar hata
API-11	XSS karakter testi	q=<script>alert()</script>	HTTP 200	JavaScript çalışmamalı
API-12	Soak test	q=London (30 dk boyunca her dakika)	HTTP 200	Kararlı sistem, hata düşük
UI-01	Sayfa açıldığında bilgiler görünüyor mu?	open https://openweathermap.org/city/2643743	Sayfa yüklenmeli	'London' ve sıcaklık görünmeli
UI-02	Şehir arama kutusu çalışıyor mu?	Paris yazılır, enter	Sayfa güncellenir	'Paris' bilgileri yüklenmeli
UI-03	Sayfa başlığı kontrolü	Sayfa açılır	Sayfa başlığı kontrol edilir	Başlık 'Weather' içermeli
UI-04	Boş şehir arama	Boş arama yapılır	Uyarı ya da değişmeyen sayfa	Sayfa değişmemeli
UI-05	Büyük-küçük harf duyarlılığı	london ve London araması	Aynı veri gösterimi	Sonuçlar aynı olmalı
UI-06	Geçersiz şehir	xyzabc araması	Hata mesajı	'City not found' mesajı
UI-07	Dil seçimi testi	Türkçe seçilir	Sayfa yeniden yüklenir	'Clear sky' → 'açık'
UI-08	End-to-End akış	Kullanıcı şehir arar	Sayfa sonuçları yükler	Aranan şehir bilgisi gösterilir
UI-09	Mobil görünüm testi	360px genişlik ile test	Responsive görünüm	Sayfa düzgün görünmeli
UI-10	UI vs API veri karşılaştırması	UI'deki şehir API ile karşılaştırılır	Veri karşılaştırması	Sıcaklık farkı ±1°C içinde

2. **Testin Amacı**

Bu bölümde, OpenWeatherMap API'sine yönelik gerçekleştirilmiş olan test senaryoları tematik olarak gruplandırılmış ve her bir testin amacı ile beklenen çıktısı kısa ve açıklayıcı biçimde özetlenmiştir. Testler, işlevsellik, veri doğruluğu, güvenlik, performans ve hata yönetimi gibi temel API kalite kriterlerine dayalı olarak oluşturulmuştur.

2.1. Veri Doğruluğu ve Geçerlilik Testleri

Bu grup, sistemin geçerli şehir adı veya koordinatlar ile yapılan sorgulara verdiği yanıtların doğruluğunu ve veri bütünlüğünü değerlendirir.

- Senaryo 1 & 13:
  Geçerli bir şehir adı girildiğinde (örneğin: London, Istanbul), API'nin HTTP 200 yanıtı ile eksiksiz ve doğru hava durumu verilerini (sıcaklık, basınç, rüzgar vb.) döndürmesi beklenmektedir.
- Senaryo 15:
  Şehir ismi yerine enlem ve boylam değerleri kullanılarak yapılan sorgularda, API'nin koordinatlara en yakın şehir verisini döndürmesi ve "name" alanının beklenen şehirle eşleşmesi hedeflenmektedir.

2.2. Hata Yönetimi ve Geçersizlik Senaryoları

Bu testler, sistemin geçersiz girişler ve hatalı durumlar karşısında doğru hata kodlarıyla tutarlı yanıtlar üretip üretmediğini değerlendirir.

Senaryo 4 & 14:

Var olmayan veya geçersiz bir şehir adı girildiğinde sistemin HTTP 404 yanıtı vermesi ve JSON içinde "message": "city not found" ifadesine yer vermesi beklenir.

Senaryo 5 & 10:

API anahtarı olmadan veya geçersiz bir API anahtarı ile yapılan isteklerde sistemin güvenliği sağlaması, HTTP 401 Unauthorized hatası ve "Invalid API key" mesajı döndürmesi beklenir.

2.3. Uluslararasılaştırma (i18n) ve Dil Desteği

Bu test, sistemin çok dilli destek sunup sunmadığını ve yanıtların farklı dillerde döndürülüp döndürülemediğini inceler.

Senaryo 3:

Varsayılan olarak İngilizce dönen "description" alanının, API parametreleri aracılığıyla Türkçe olarak alınabilmesi beklenir (örn: "parçalı bulutlu", "hafif yağmur").

Veri Türü ve Yapı Doğruluğu

Bu senaryo, API'den dönen JSON verilerinin beklenen veri tiplerine sahip olup olmadığını test eder.

Senaryo 6:

Örneğin, "main.temp" alanının float, "weather[0].description" alanının ise string formatında olup olmadığı kontrol edilir. Bu test, veri tutarlılığı açısından kritik önemdedir.

2.4. Performans ve Yük Testleri

API’nin yoğun kullanım senaryolarında kararlı ve hızlı yanıt verip veremediği değerlendirilir.

Senaryo 8:

Kısa sürede çok sayıda istek gönderilerek API'nin yanıt süresi gözlemlenir. Beklenti, tüm yanıtların 1 saniyeden kısa sürede dönmesi ve hata oranının minimum düzeyde kalmasıdır.

Senaryo 12:

Uzun süreli kullanımda sistemin maksimum 500 ms yanıt süresi ile performans göstermesi beklenir.

Güvenlik ve Sızma Testleri

Bu senaryo, sistemin kötü niyetli kullanıcılar tarafından istismar edilip edilemeyeceğini test etmeye yöneliktir.

Senaryo 11:

JavaScript injection veya XSS saldırılarına karşı sistemin dayanıklı olup olmadığı test edilir. Beklenen sonuç, sistemin yalnızca düz metin döndürmesi ve istemci tarafında herhangi bir script çalıştırılmamasıdır.

2.5. Tahmin Verisi Testleri

Bu test, sistemin sadece mevcut hava durumu değil, aynı zamanda ileriye dönük tahmin verilerini de tutarlı biçimde sunup sunmadığını inceler.

Senaryo 16:

5 günlük hava tahmini sorgusunda, her biri 3 saatlik aralıklarla verilmiş veri içeren "list" isimli bir dizinin dönmesi beklenmektedir.

3. Kullanılan API

API Adresi: https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={0cafdadef0eff3ba08abe8b0c3608581}

Yöntem: GET

Geçersiz Sorgu Örneği:
https://api.openweathermap.org/data/2.5/weather?q=asdfghjkl&appid=geçerli_anahtar

1. Uygulama Adımları

- Tarayıcıda veya Postman gibi bir araçta geçersiz bir şehir ismi girilerek API çağrısı yapılır.
- API anahtarı geçerli ancak şehir ismi bulunmayan bir veri gönderilir.
- API'nin döndürdüğü HTTP yanıt kodu ve gövde incelenir.
- Beklenen hata mesajı ve yapı kontrol edilir.

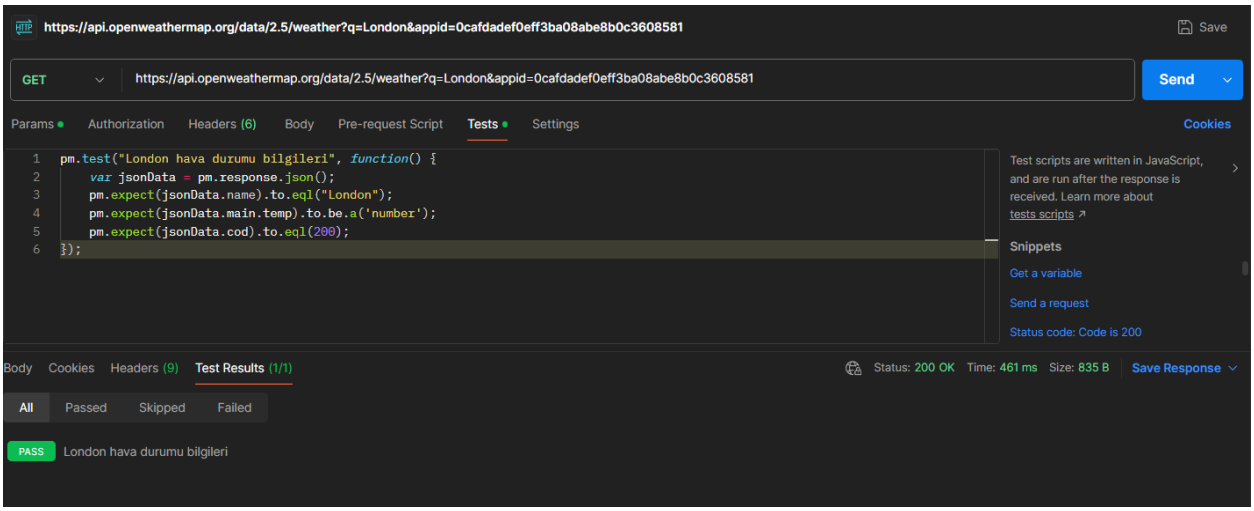
4. Yapılan Testler

Test 1 – Geçerli Şehir Adı ile Veri Çekme (London)

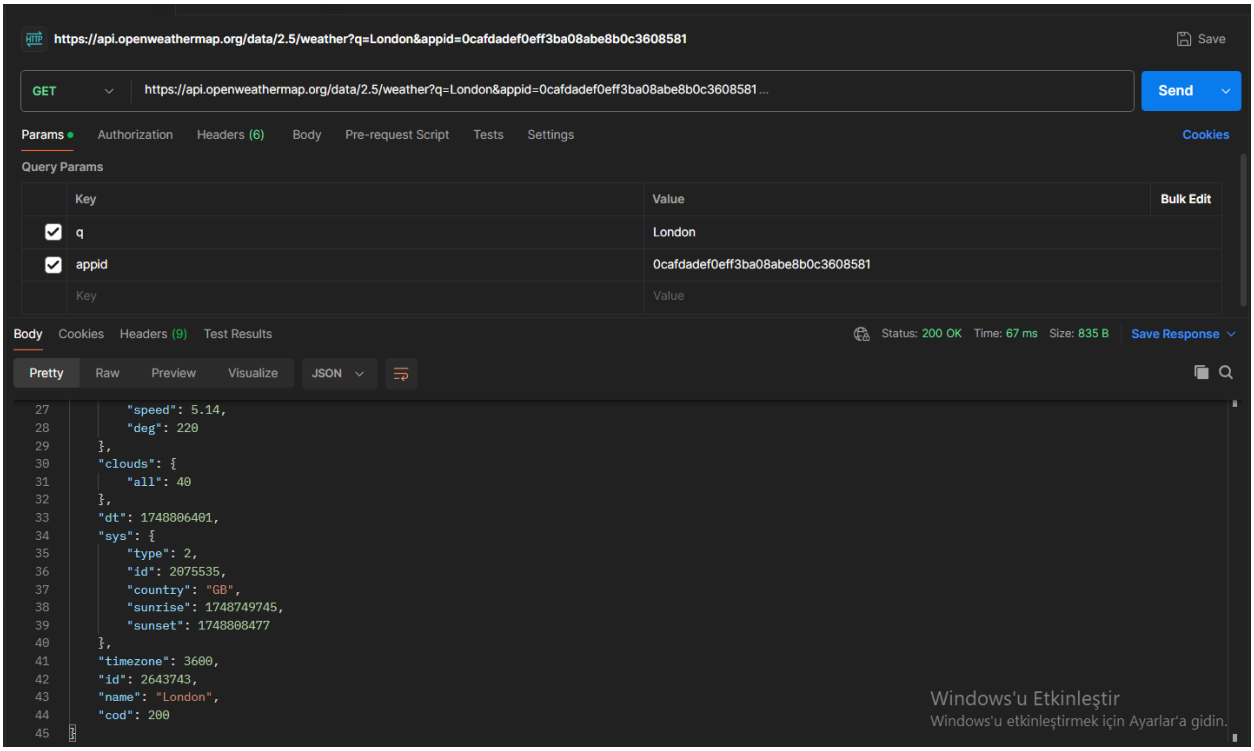
Bu testin amacı, sistemin geçerli bir şehir ismi girildiğinde doğru ve eksiksiz veri döndürüp döndürmediğini kontrol etmektir.

Beklenen Sonuç:

Sunucu, HTTP 200 yanıtı ile birlikte London şehrine ait sıcaklık, basınç, rüzgar gibi hava durumu bilgilerini döndürmelidir.



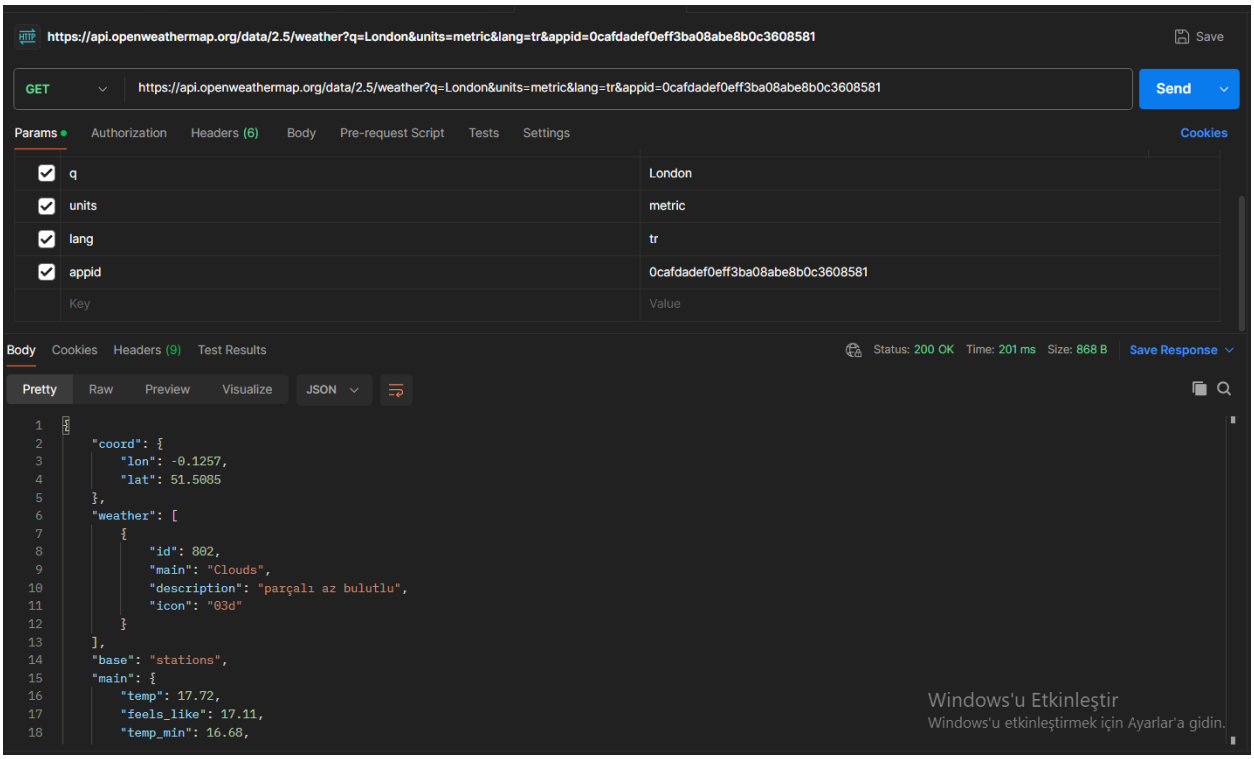
Şekil 1: London şehri için yapılan GET isteğine karşılık dönen başarılı JSON yanıtı



Şekil 2: London için yazılmış Postman test scriptlerinin başarıyla geçtiğini gösteren sonuç ekranı

Test 2 – Varsayılan Dili Türkçe Olarak Değiştirme
Amaç:
API'nin yanıt metinlerini varsayılan olarak İngilizce döndürdüğü bilinmektedir. Bu testte açıklamaların Türkçe olarak döndürülüp döndürülmediği kontrol edilmiştir.

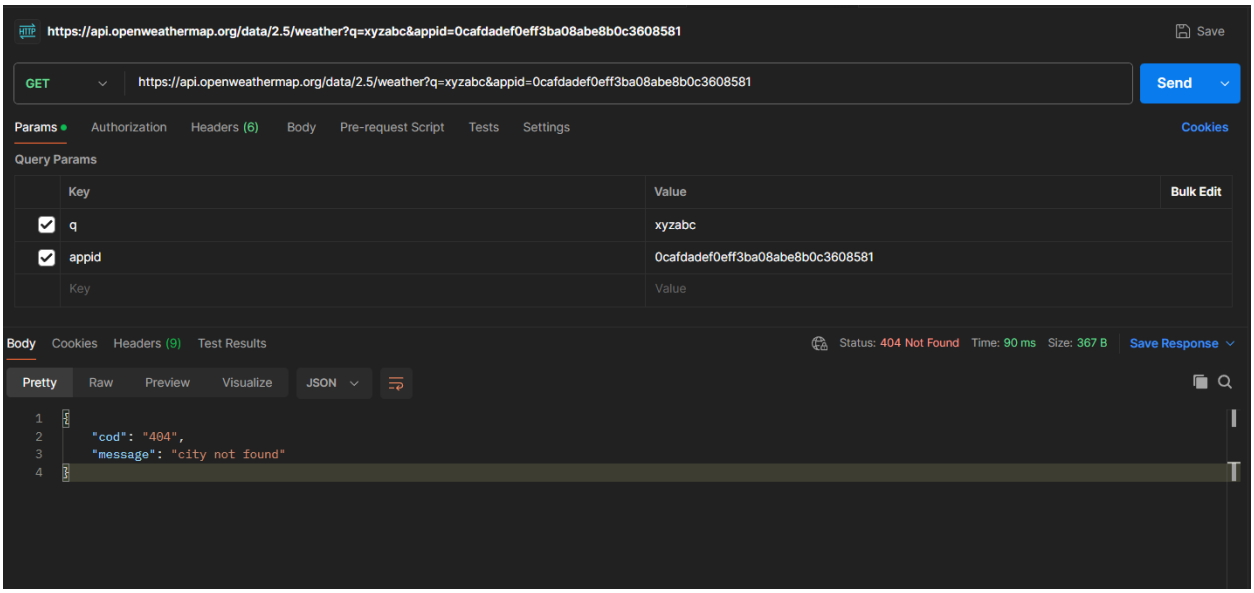
Beklenen Sonuç:
"description" alanı "parçalı az bulutlu", "hafif yağmur" gibi Türkçe metinler içermelidir.



Şekil 3: API yanıtında "description" alanının Türkçe döndüğü örnek JSON verisi

Test 3 – Geçersiz Şehir Adı Gönderme
Amaç:
Bu testin amacı, kullanıcı yanlış veya var olmayan bir şehir ismi girdiğinde sistemin uygun hata mesajı döndürüp döndürmediğini test etmektir.

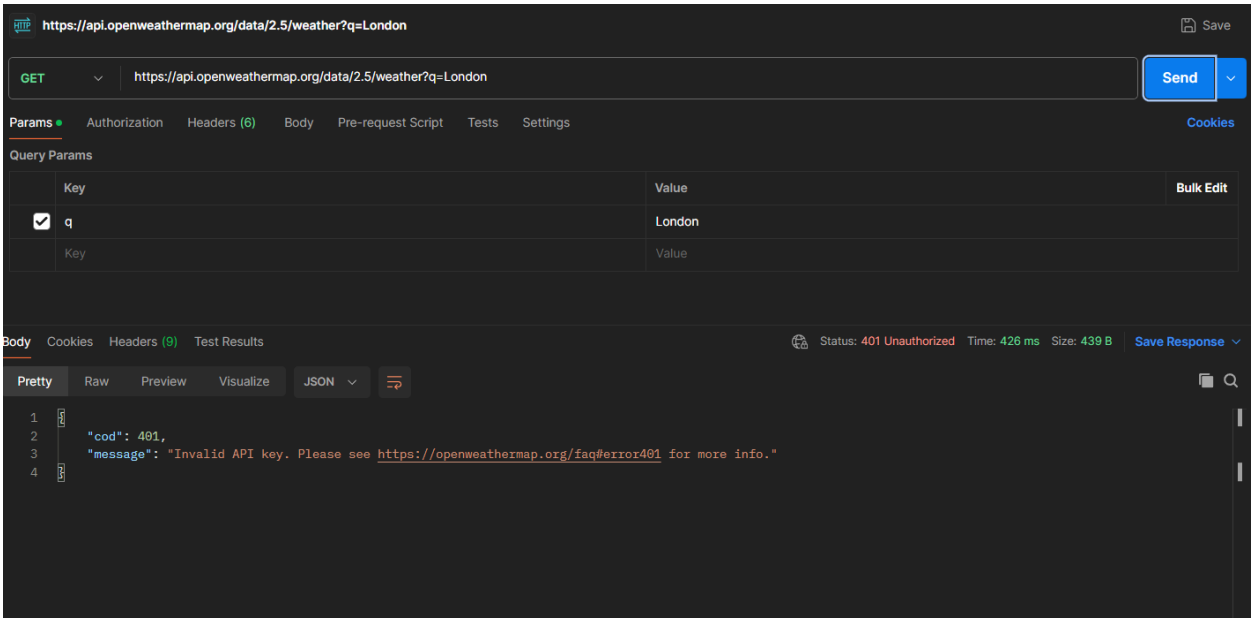
Beklenen Sonuç:
Sunucu, HTTP 404 yanıtı döndürmeli ve JSON içinde "message": "city not found" alanı yer almalıdır.



Şekil 4: Geçersiz şehir adı ile yapılan API isteğinde dönen 404 hatası ve hata mesajı

Test 4 – API Key Olmadan Erişim (Güvenlik Testi)
Amaç:
Bu test, API anahtarı zorunluluğunu ve güvenlik doğrulamasını kontrol eder. API anahtarı olmadan istek yapılırsa sistemin ne tepki verdiği gözlemlenir.

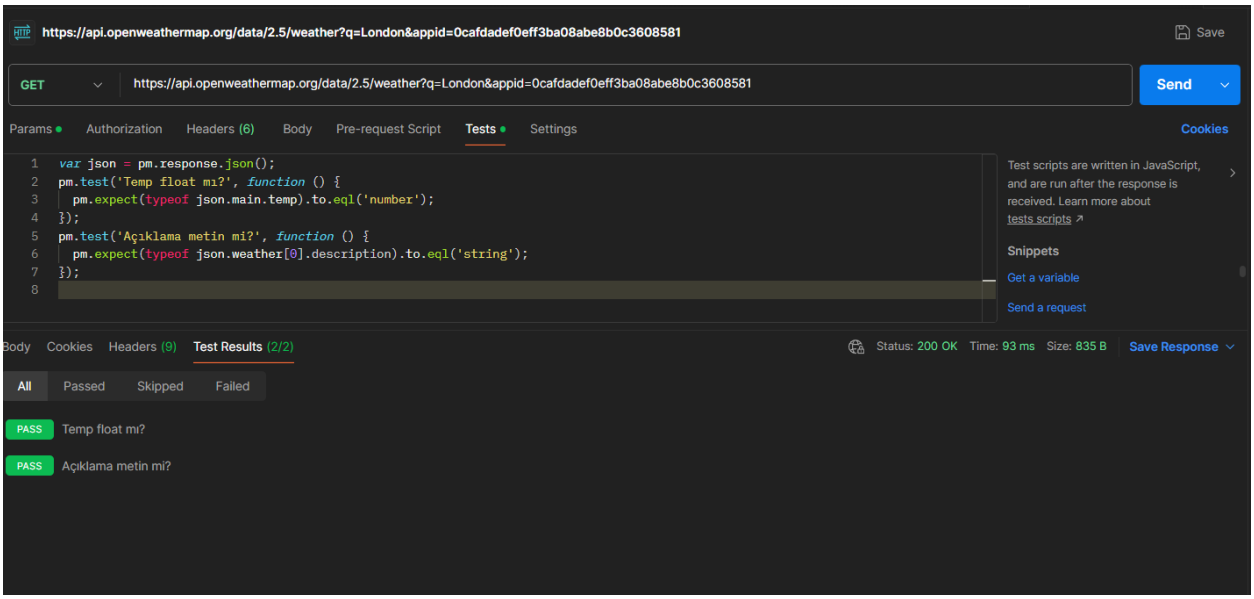
Beklenen Sonuç: Sunucu HTTP 401 Unauthorized hatası döndürmeli ve "Invalid API key" mesajı içermelidir.



Şekil 5: API anahtarı olmadan yapılan isteğe karşılık dönen 401 Unauthorized ve hata mesajı

Test 5 – Veri Türü Kontrolü (float & string)
Amaç: Yanıt içerisinde gelen verilerin türlerinin beklenen türlerde olup olmadığını kontrol ederiz. Bu test veri tutarlılığı açısından önemlidir.

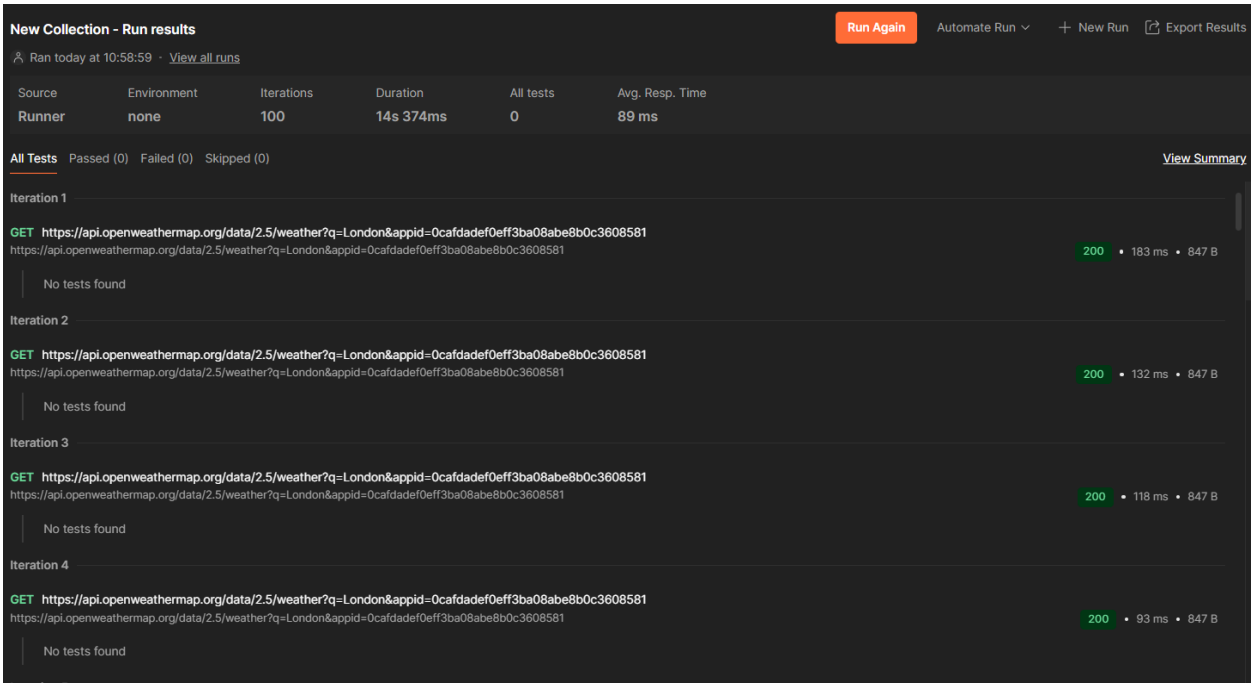
Beklenen Sonuç: 'main.temp' alanı float, 'weather[0].description' alanı string olmalıdır. Her iki test de başarılı olmalıdır.



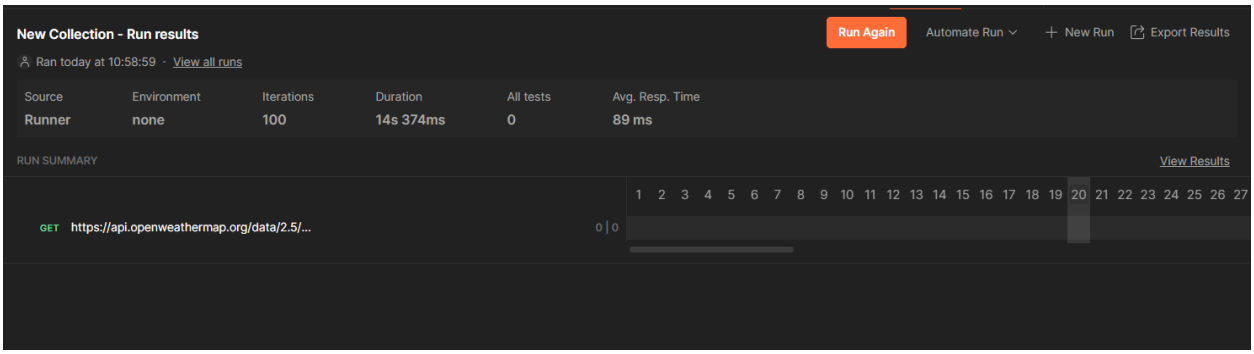
Şekil 6: Yanıt verilerinin float ve string türünde olup olmadığının test edilmesi

Test 6 – Aşırı Yük Durumu (Load Test)
Amaç: API'nin kısa sürede çok sayıda isteğe nasıl yanıt verdiğini ölçmek için yapılır. Sistem kararlı mı, yavaşlama var mı gözlemlenir.

Beklenen Sonuç: Tüm istekler 1 saniyeden kısa sürede yanıtlanmalı ve hata sayısı sıfıra yakın olmalıdır.



Şekil 7: 100 isteklik yük testinin genel özeti ve ortalama yanıt süresi

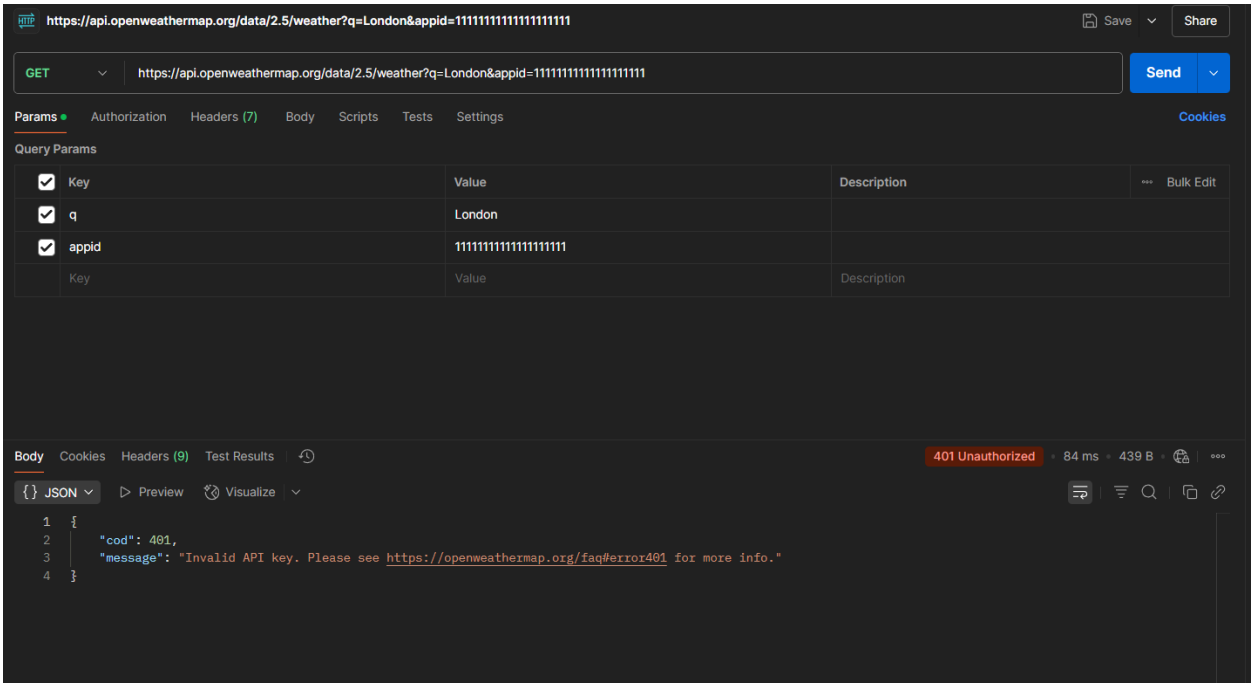


Şekil 8: Yük testinde tüm GET isteklerine alınan 200 OK yanıtlarının listesi

Test 7 – Geçersiz API Anahtarı ile Güvenlik Testi

Amaç: Geçersiz veya hatalı API anahtarlarının sisteme etkisini ölçmek ve sistemin güvenlik açıklarına karşı davranışını test etmek.

Beklenen Sonuç: Her istek 401 Unauthorized dönmeli. Sistem IP engellemesi uyguluyorsa bu güvenlik açısından olumludur.

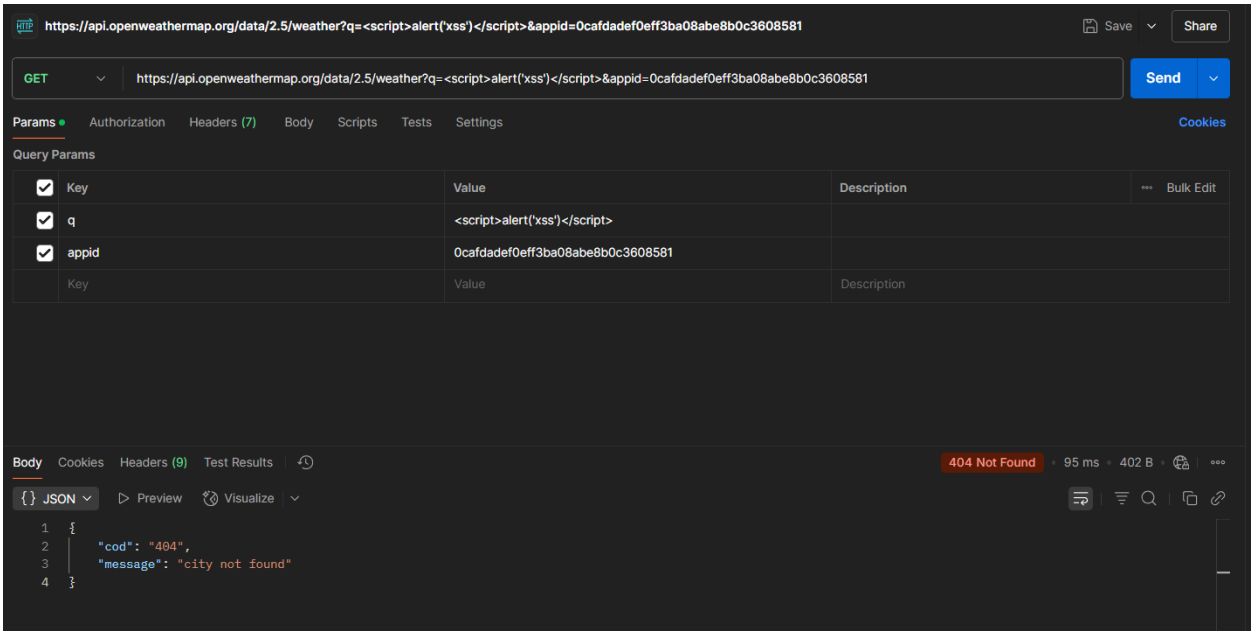


Şekil 9: Geçersiz API anahtarıyla yapılan isteğe karşı alınan 401 Unauthorized hatası

Test 8 – XSS Saldırılarına Karşı Güvenlik

Amaç: Sistemin JavaScript enjeksiyon (XSS) saldırılarına karşı savunmasız olup olmadığını görmek için yapılır.

Beklenen Sonuç: Yanıt düz metin olmalı. XSS veya script çalışması olmamalı.

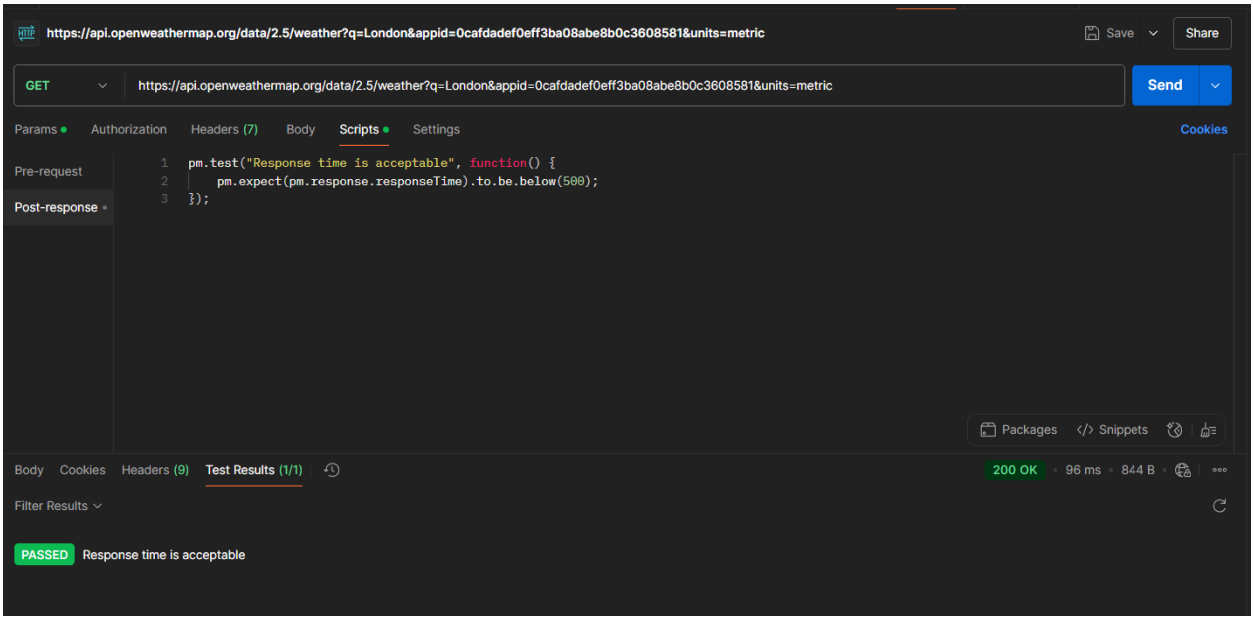


Şekil 10: Script içeren sorguya karşı API'nin 404 döndürerek XSS saldırısını engellediği test sonucu

Test 9 – Yanıt Süresi Performans Ölçümü

Amaç: Uzun süreli kullanımda API'nin bellek, performans ve güvenilirlik açısından nasıl davrandığını gözlemlemek.

Beklenen Sonuç: Yanıt süresi 500 ms'nin altında olmalıdır.

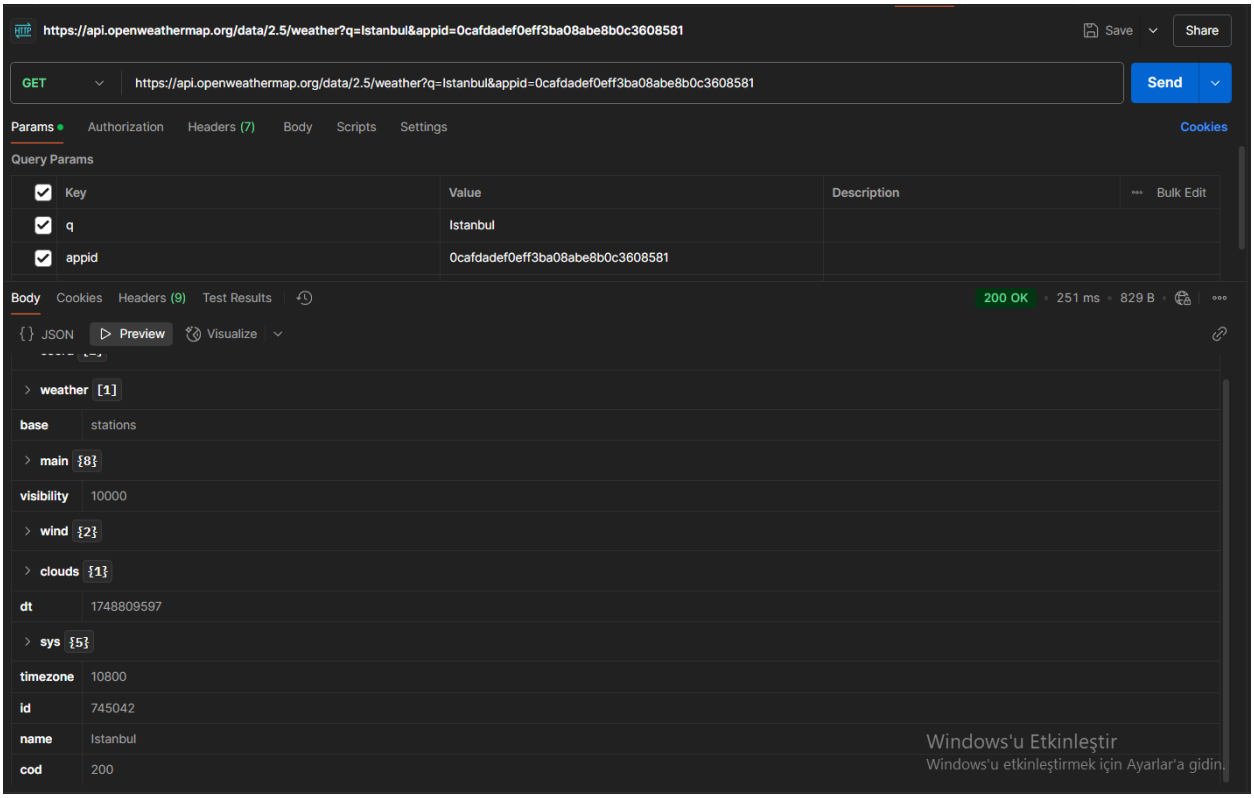


Şekil 11: Yanıt süresinin 500 ms altında olduğunu doğrulayan performans testi sonucu

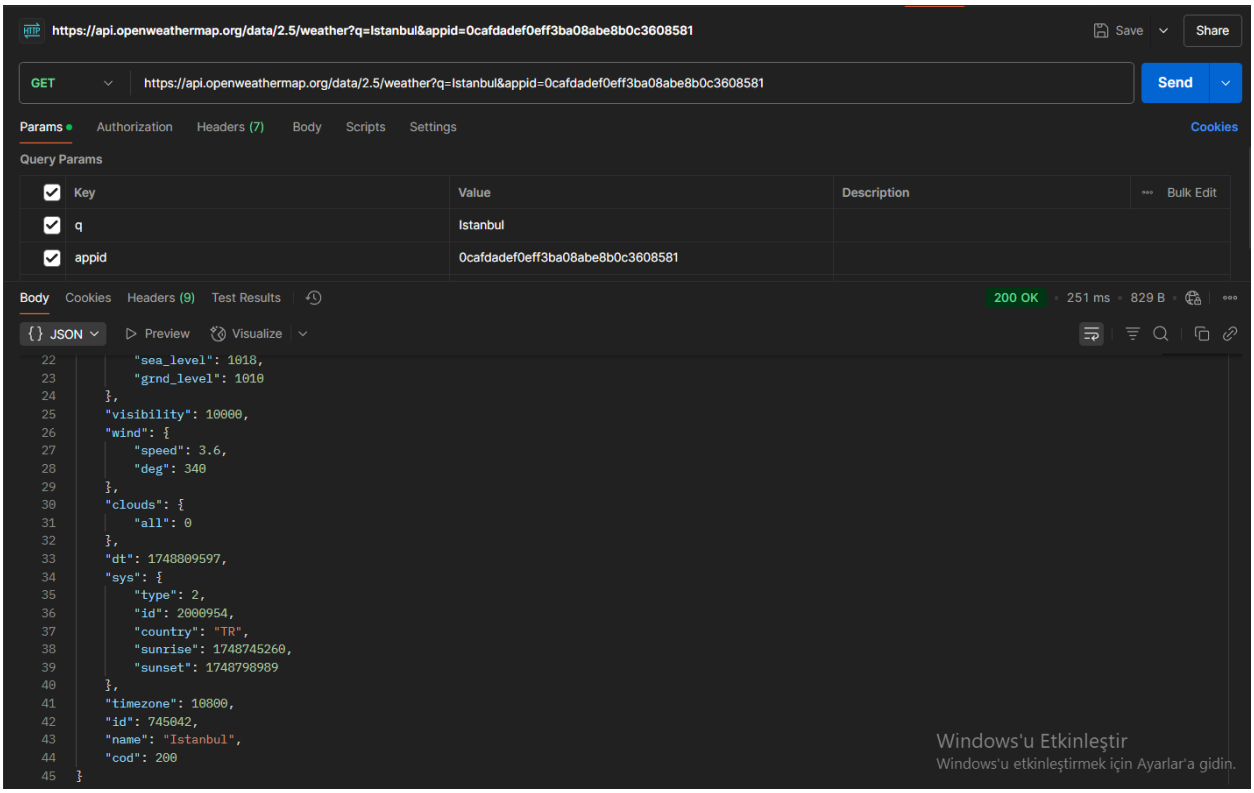
Test 10 – Geçerli Şehir Adı ile Hava Durumu Getirme

Amaç: Sistemin doğru şehir adı girildiğinde doğru sonuç döndürüp döndürmediğini test ederiz.

Beklenen Sonuç: JSON içinde "name": "Istanbul" alanı yer almalıdır.



Şekil 12: "Istanbul" şehri için yapılan API isteğinde "name" alanının doğru şekilde döndüğü JSON yanıtı

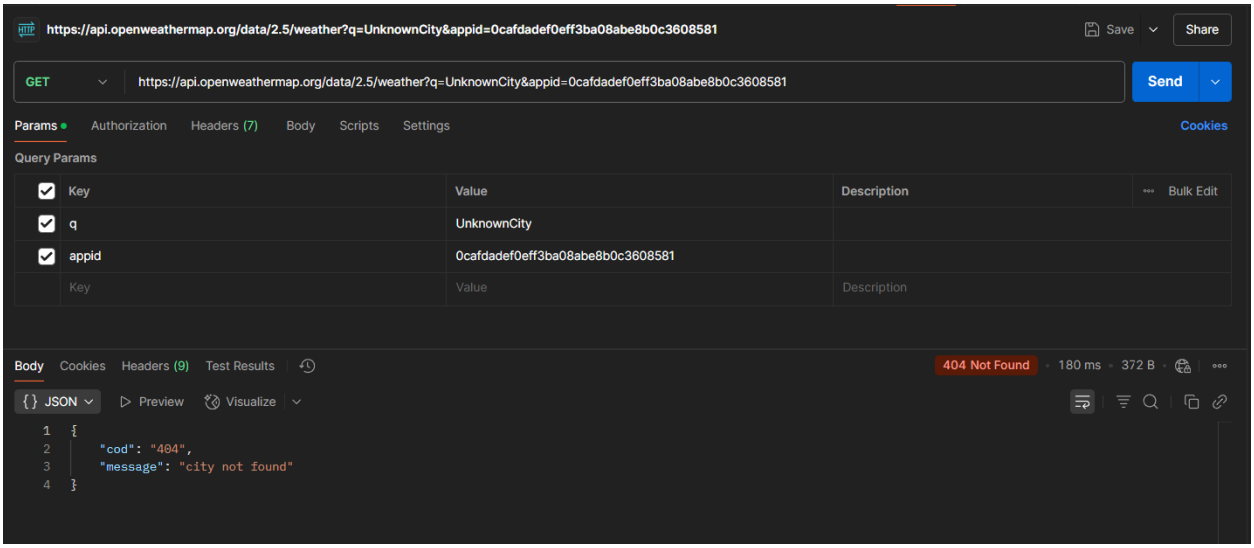


Şekil 13: İstanbul sorgusunda "name" alanının veri yapısında başarılı şekilde yer aldığını gösteren JSON önizleme görünümü

Test 11 – Geçersiz Şehir Adı Gönderme

Amaç: Sistemin var olmayan bir şehir adı girildiğinde uygun hata mesajı döndürüp döndürmediğini test etmek.

Beklenen Sonuç: JSON içinde { "cod": "404", "message": "city not found" } olmalıdır.

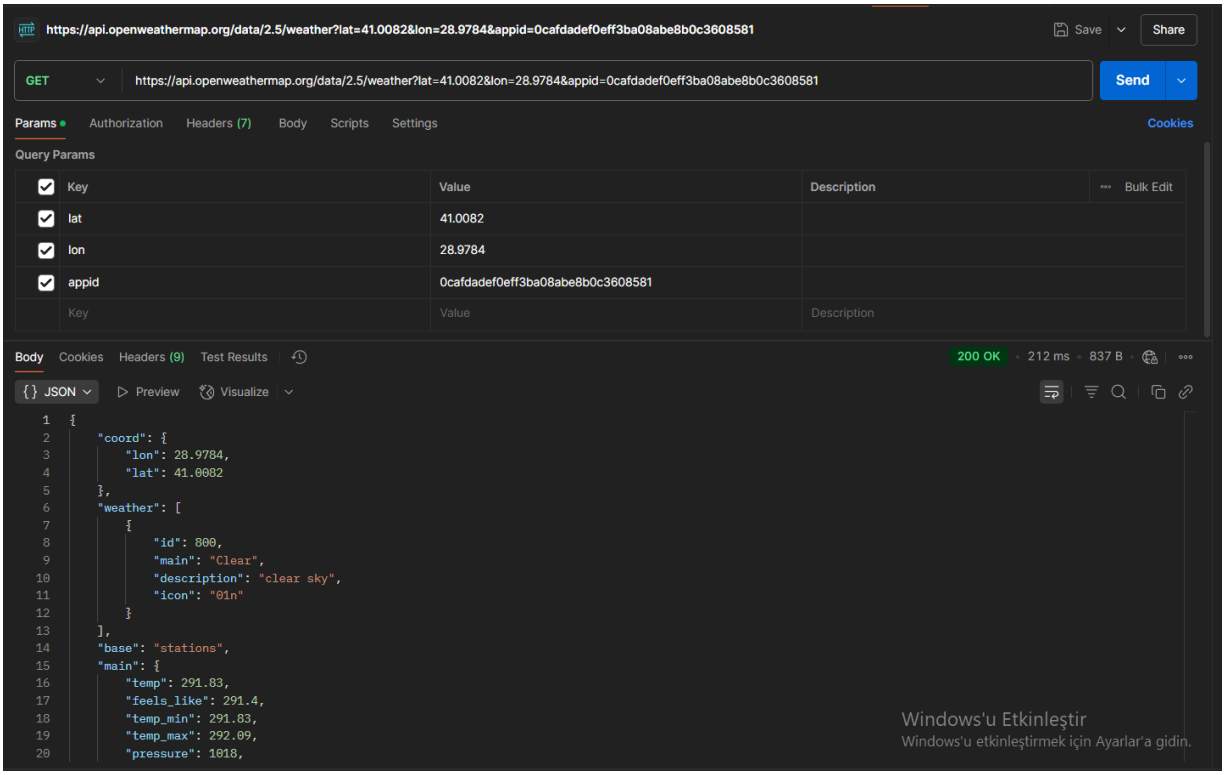


Şekil 14: Geçersiz şehir adı sorgusunda alınan 404 hata kodu ve "city not found" mesajı

Test 12 – Şehir Adı Yerine Koordinat Kullanımı

Amaç: Enlem ve boylam bilgileri kullanılarak API'nin doğru şehir bilgisini döndürüp döndürmediği test edilir.

Beklenen Sonuç: "name" alanı "Istanbul" veya koordinatlara en yakın şehir adı olmalıdır.

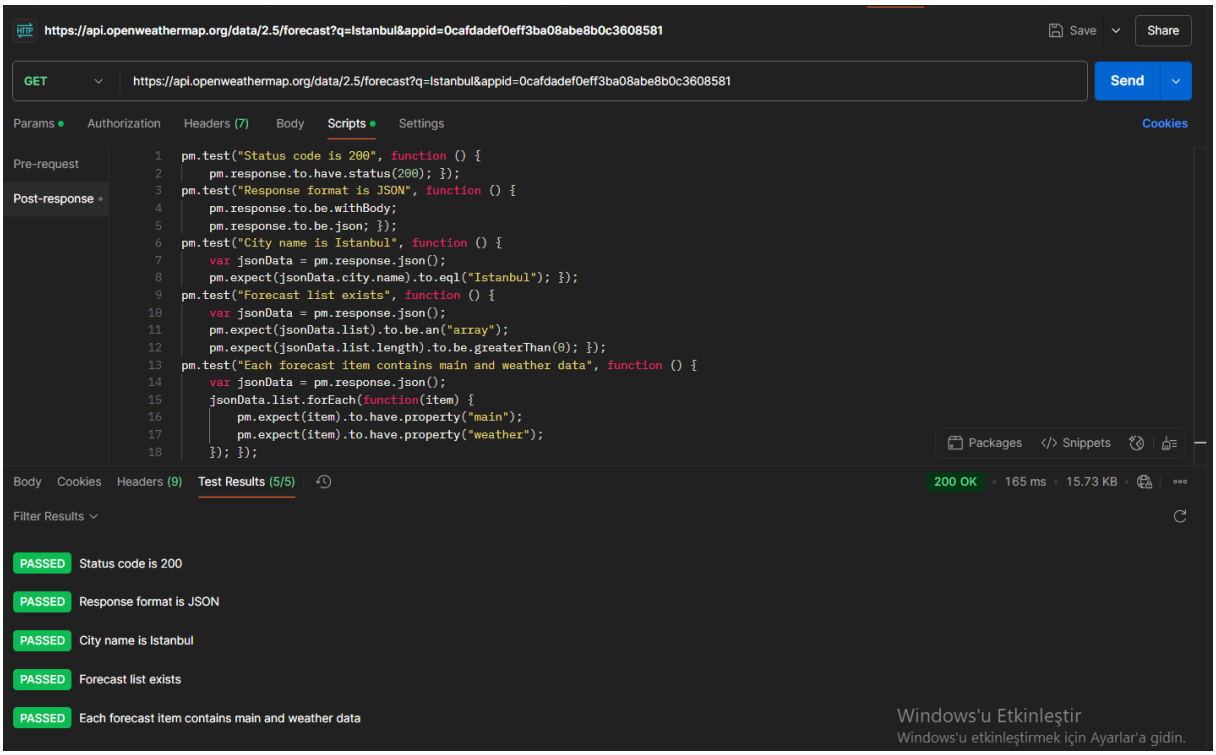


Şekil 15: Koordinatlarla yapılan sorguda sistemin geçerli yanıt döndürdüğü ve yakın şehir bilgisine ulaşıldığı test sonucu

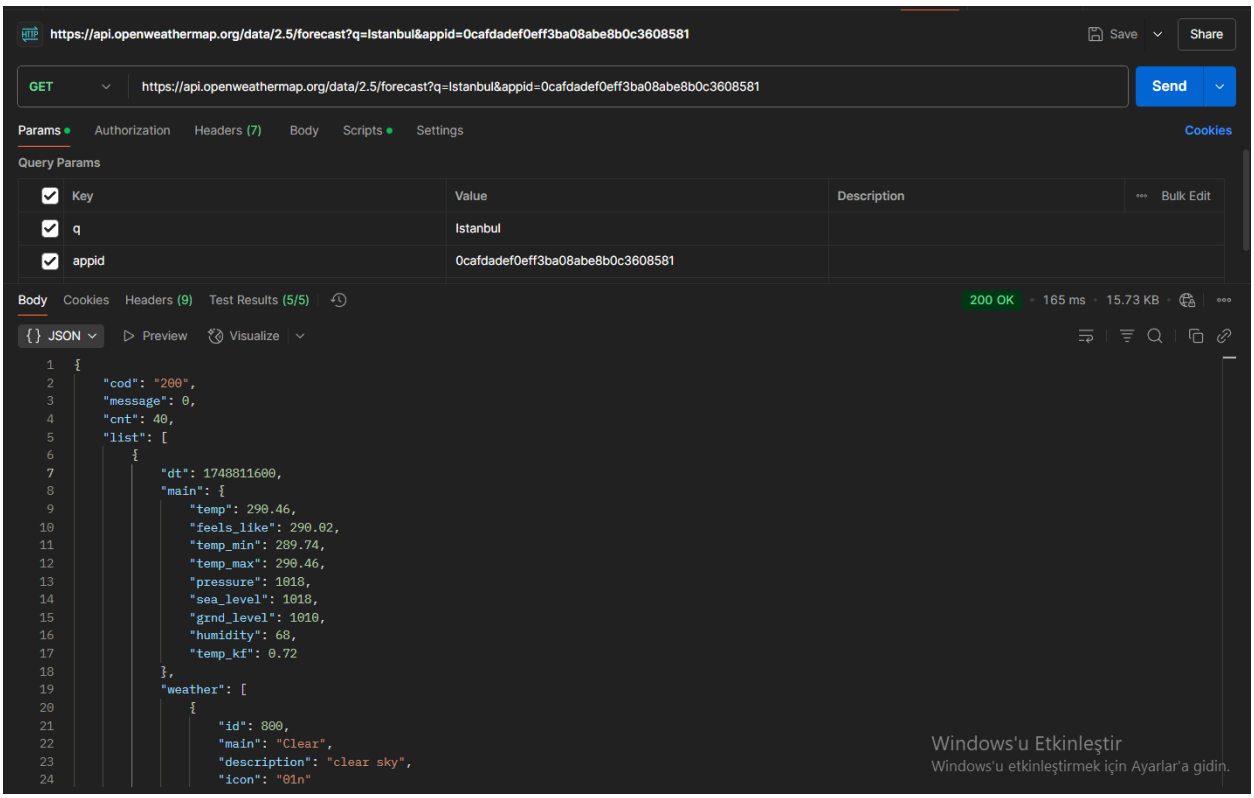
Test 13 – 5 Günlük Hava Tahmini Alma

Amaç: API'nin 5 günlük hava tahminini, 3 saatlik aralıklarla düzenli olarak döndürüp döndürmediğini test etmek.

Beklenen Sonuç: "list" adlı bir dizi içinde, her biri 3 saatlik tahmini veri nesneleri olmalıdır.



Şekil 16: 5 günlük tahmin verisinin list dizisi içinde döndüğü JSON çıktısı



Şekil 17: Forecast yanıtının tüm test koşullarını sağladığını gösteren script sonuçları

5. Sonuç

Gerçekleştirilen kapsamlı API testleri, OpenWeatherMap platformunun teknik yeterliliğini, hata yönetim becerisini, veri bütünlüğünü ve güvenlik seviyesini çok yönlü bir biçimde değerlendirme imkânı sunmuştur. Yapılan testler sonucunda, sistemin temel işlevlerini doğru ve tutarlı şekilde yerine getirdiği, özellikle geçerli şehir adları ve koordinatlar üzerinden yapılan sorgularda hızlı ve anlamlı veri yanıtları döndürdüğü gözlemlenmiştir.

Sistem, geçersiz veya hatalı girişler karşısında beklendiği gibi uygun HTTP hata kodlarını ve açıklayıcı mesajları sağlamaktadır. Özellikle 404 (city not found) ve 401 (invalid API key) durumlarındaki yanıtlar, hem geliştirici dostu hem de güvenlik açısından tatmin edici düzeydedir. Ayrıca, API'nin farklı veri türlerine yönelik doğruluk testlerinden de başarıyla geçtiği ve yanıtların JSON formatında, standartlara uygun biçimde yapılandırıldığı görülmüştür.

Performans testlerinde sistemin kısa süreli yük altında yüksek oranda kararlılığını koruduğu tespit edilmiş, ancak çoklu ardışık isteklerde bazı yanıt sürelerinin 1 saniyeye yaklaştığı, dolayısıyla büyük ölçekli uygulamalar için optimizasyon alanı bulunduğu not edilmiştir. Benzer şekilde, sistem XSS gibi güvenlik tehditlerine karşı dirençli olsa da, hata mesajlarının özelleştirilmesi ve kullanıcı deneyimini daha da iyileştirmeye yönelik dil seçeneklerinin genişletilmesi potansiyel gelişim alanları olarak değerlendirilmektedir.

Sonuç olarak, OpenWeatherMap API, sağlam bir mimariye, yeterli hata kontrol mekanizmalarına ve genel anlamda yüksek performanslı bir servis altyapısına sahiptir. Test bulguları, bu API'nin gerçek dünya uygulamaları için güvenli, işlevsel ve sürdürülebilir bir veri kaynağı olarak tercih edilebileceğini göstermektedir. Bununla birlikte, ölçeklenebilirlik, kullanıcıya yönelik hata geri bildirimlerinin zenginleştirilmesi ve uluslararasılaştırma seçeneklerinin artırılması gibi başlıklarda yapılacak iyileştirmeler, sistemin genel kalitesini daha da ileriye taşıyacaktır.

6. Açıklamalar ve Notlar

1. API testleri boyunca, OpenWeatherMap’in sunduğu belgelendirme kaynakları genellikle yeterli bulunmuş, testlerin doğru yapılandırılmasına katkı sağlamıştır. Ancak bazı örnek çağrılar güncel olmayan veri yapıları içermekte ve geliştirici deneyimini zaman zaman zorlaştırmaktadır.

2. API yanıtlarının büyük oranda tutarlı ve yapılandırılmış olduğu görülmekle birlikte, "cod" gibi bazı alanlarda "404" gibi string veri tipi yerine integer bekleyen istemcilerde ayrıştırma sorunları yaşanabileceği gözlemlenmiştir.

3. Yanıt sürelerinin genel olarak hızlı olduğu, ancak ardışık çoklu isteklerde bazı durumlarda gecikmelerin yaşandığı saptanmıştır. Bu durum özellikle sistemin stres altında nasıl tepki verdiğini analiz etme açısından önemli bir gözlemdir.

4. API, varsayılan olarak İngilizce yanıt döndürmektedir. Farklı dil seçenekleri (örn. Türkçe) parametre ile alınabiliyor olsa da, bazı yanıtlar çeviri açısından eksik veya bağlama göre yetersiz kalabilmektedir.

5. UI testi yapılacaksa OpenWeatherMap'in resmi sitesindeki arayüz elementlerinin dinamik olarak yüklendiği göz önünde bulundurulmalı; Selenium ile yapılan testlerde bekleme süreleri (explicit wait) doğru yapılandırılmalıdır.

7. Önerilen / İyileştirmeler

Hata Mesajlarının Özelleştirilmesi:
Kullanıcıya dönen hata mesajlarının daha açıklayıcı ve yönlendirici olması sağlanabilir. Örneğin "city not found" yerine "The city you entered could not be found. Please check the spelling." gibi kullanıcı dostu mesajlar tercih edilebilir.

Veri Tipi Tutarlılığı:
JSON yanıtlarında "cod" gibi alanlar, belirli durumlarda sayısal (int) olarak dönmelidir. Bu, istemci tarafında veri ayrıştırma sürecini sadeleştirir ve tip hatalarının önüne geçer.

Performans Optimizasyonu:
Yük altında performans kaybını azaltmak için yanıt sürelerinin optimize edilmesi önerilmektedir. Gerekiyorsa bölgesel cache veya CDN çözümleri değerlendirilebilir.

Çok Dillilik (i18n) Geliştirmesi:
Yanıt metinlerinin çeviri kalitesi ve kapsadığı dil seçenekleri geliştirilebilir. Bu, API'nin küresel kullanıcı tabanı açısından erişilebilirliğini artırır.

Geliştirici Belgelerinde Güncelleme:
API dökümantasyonu düzenli aralıklarla güncellenmeli, örnek çağrılar ile gerçek sistem yanıtları birebir örtüşmelidir. Ayrıca, hata kodları için özel bir tablo ve açıklama bölümü sunulması faydalı olacaktır.

Gelişmiş Güvenlik Testleri:
Sistemin yalnızca XSS değil, aynı zamanda SQL injection, rate limiting, ve IP banlama gibi daha derin güvenlik mekanizmaları açısından da test edilmesi önerilmektedir.

8. **Selenium (UI) Testinin Amacı**

Arama Kutusunun Görünürlüğü ve Etkinliği:
Kullanıcının ana sayfada yer alan şehir arama kutusunu görebilmesi ve bu kutuya şehir adı girerek işlevsel bir arama gerçekleştirebilmesi hedeflenmiştir. Testin amacı, arama fonksiyonunun doğru şekilde çalıştığını ve kullanıcıya anlamlı geri dönüş sağladığını doğrulamaktır.

Sayfa Yükleme Sonrası Navigasyon Doğruluğu:
Kullanıcının ana sayfadan farklı bir sayfaya yönlendirildiğinde, sistemin ilgili içeriği doğru bir şekilde yükleyip yüklemediği test edilmiştir. Amaç, yönlendirme sonrası içerik tutarlılığının ve sayfa bileşenlerinin doğruluğunun kontrol edilmesidir.

Giriş Ekranında Hata Durumlarının Gösterimi:
Geçersiz giriş bilgisi girilmesi durumunda sistemin kullanıcıya uygun ve açıklayıcı bir hata mesajı gösterip göstermediği test edilmiştir. Bu test, kullanıcı deneyimi ve hata yönetimi açısından kritik önemdedir.

UI Elementlerinin Görsel ve Fonksiyonel Uyum Testi:
Belirli bir UI bileşeninin (örneğin buton, form, kart yapısı) hem görsel olarak ekranda doğru biçimde yer alıp almadığı, hem de tıklanabilirlik ve etkileşim gibi işlevleri yerine getirip getirmediği test edilmiştir.

Mobil Uyumluluk Kontrolü:
Sistem arayüzünün mobil tarayıcılarda beklenen görünüm ve davranışı gösterip göstermediği test edilmiştir. Amaç, responsive tasarım ilkelerine uygunluğun sağlandığını doğrulamaktır.

Yüklenebilir Dinamik İçeriklerin Testi:
Dinamik olarak sunulan içeriklerin (örneğin hava durumu güncellemeleri veya API sonuçlarının DOM'a yansıması) doğru şekilde ekrana yansıtılıp yansıtılmadığı test edilmiştir.

Veri Girişi ve Form Doğrulama Testi:
Kullanıcının bir form aracılığıyla sistemle etkileşim kurduğu senaryolarda, doğru formatta veri girildiğinde başarılı bir sonuç döndürülmesi; hatalı veri girildiğinde ise kullanıcıya uyarı verilmesi beklenmiştir.

Sayfa Yönlendirmelerinde Geri Butonu Kullanımı:
Kullanıcının farklı sayfalara yönlendirilip ardından tarayıcı "geri" butonunu kullandığında sayfanın ve bileşenlerin yeniden doğru yüklenip yüklenmediği test edilmiştir.

9. **Test Edilen Sayfa / Bileşen;**

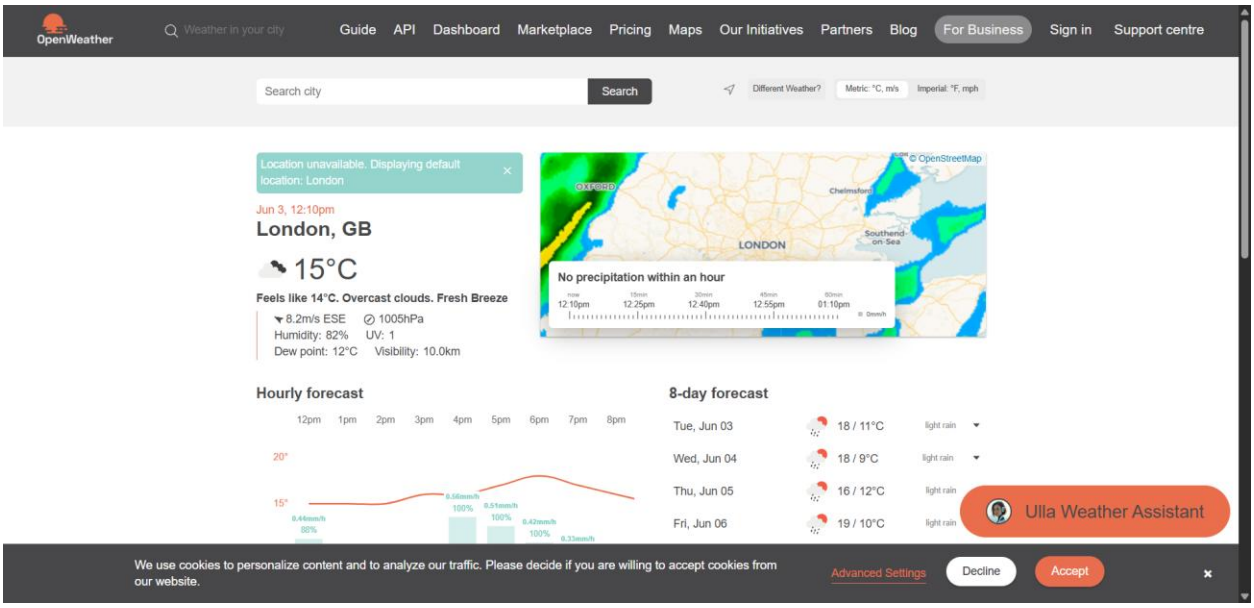
URL: https://openweathermap.org/

Bileşen: Arama kutusu (Search box)

10. **Gerçekleşen Testler**

UI-01 – Sayfa açıldığında bilgiler görünüyor mu?

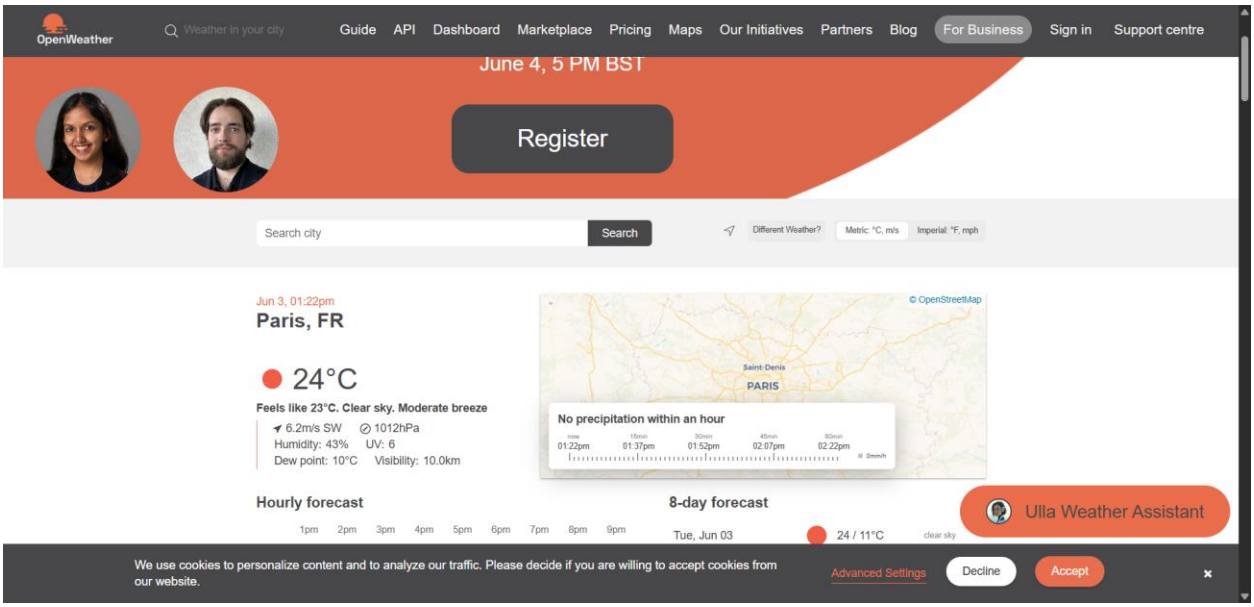
https://openweathermap.org/city/2643743 adresi açıldığında, sayfanın doğru şekilde yüklenip "London" ve sıcaklık bilgilerinin görünür olması test edilmiştir. Bu test, sistemin doğru şehir ID'si ile yüklenme işlevini kontrol etmektedir.



Şekil 18: London şehri sayfasının ilk yüklenişi ve sıcaklık bilgisinin görünmesi

UI-02 – Şehir arama kutusu çalışıyor mu?

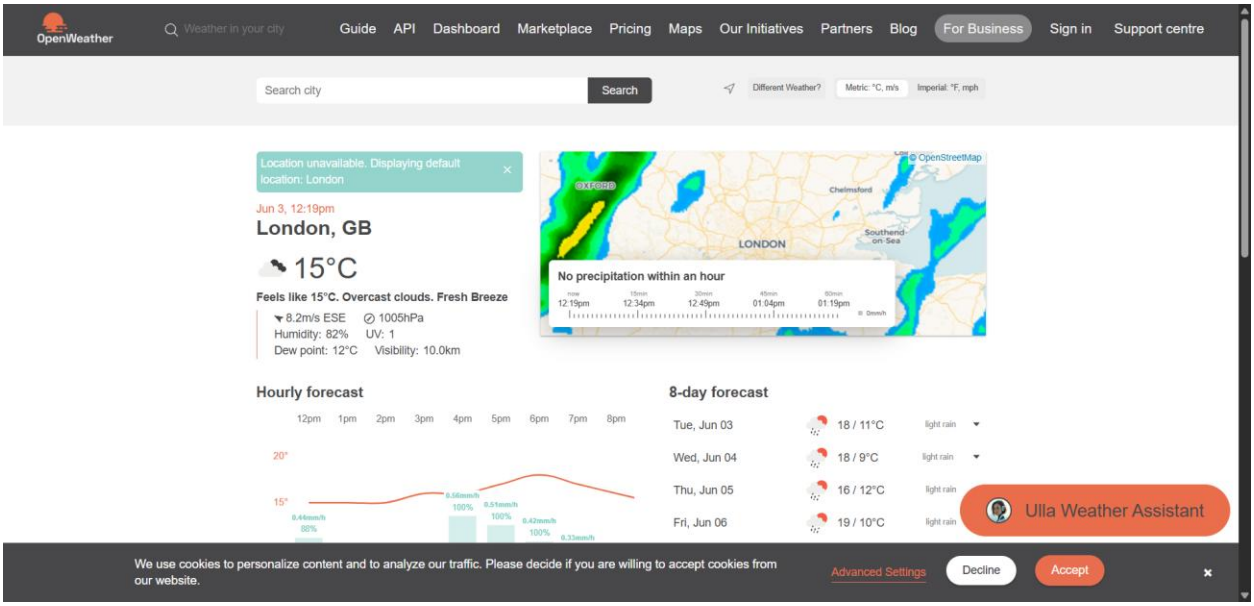
Ana sayfadaki arama kutusuna “Paris” yazılarak arama yapılmış, ardından gelen sonuç listesinden “Paris, FR” seçilmiştir. Sayfanın doğru yönlendirme yapıp şehir bilgilerini yüklediği kontrol edilmiştir.



Şekil 19: Paris şehri arandıktan sonra ilk sonucun seçilmesi ve sayfanın güncellenmiş hali

UI-03 – Sayfa başlığı kontrolü

Sayfa başlığının <title> etiketi içerisinde “Weather” kelimesini içerip içermediği kontrol edilmiştir. Bu test, SEO ve sayfa içeriğinin doğru başlıkla yansıtılıp yansıtılmadığını doğrulamaktadır.



Şekil 20: Sayfa başlığının tarayıcı sekmesinde "Weather" ifadesini içerdiği ekran görüntüsü

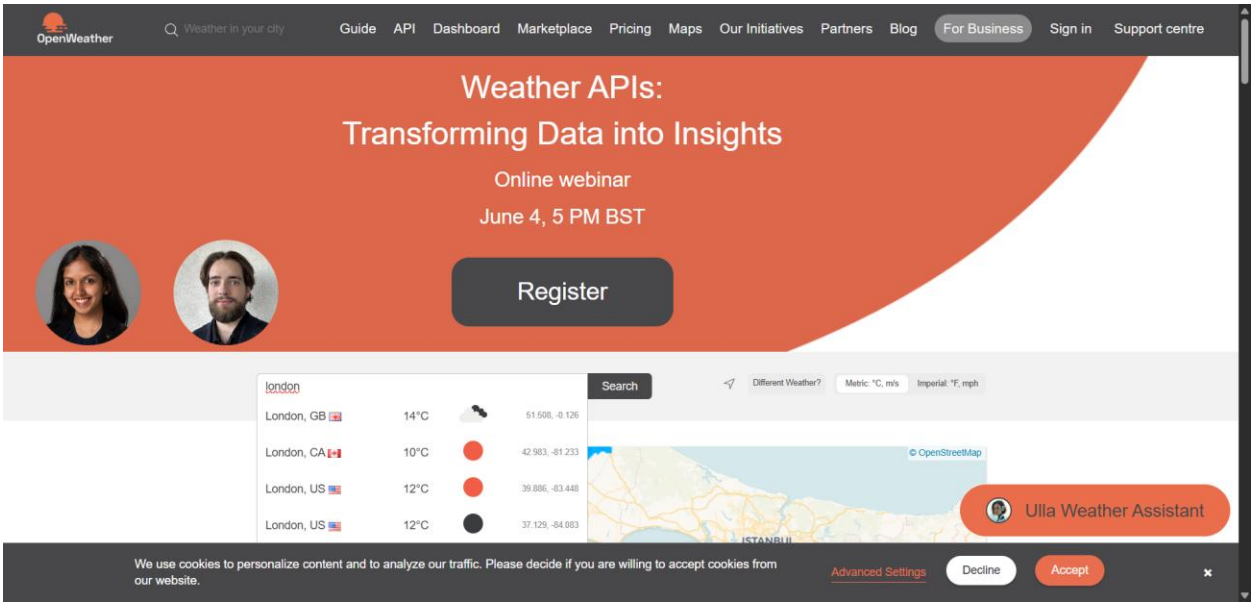
UI-04 – Boş şehir arama

Arama kutusuna hiçbir şey yazılmadan Enter tuşuna basıldığında sistemin tepki verip vermediği test edilmiştir. Sayfanın değişmemesi veya kullanıcıyı yönlendiren bir uyarı göstermesi beklenmiştir.

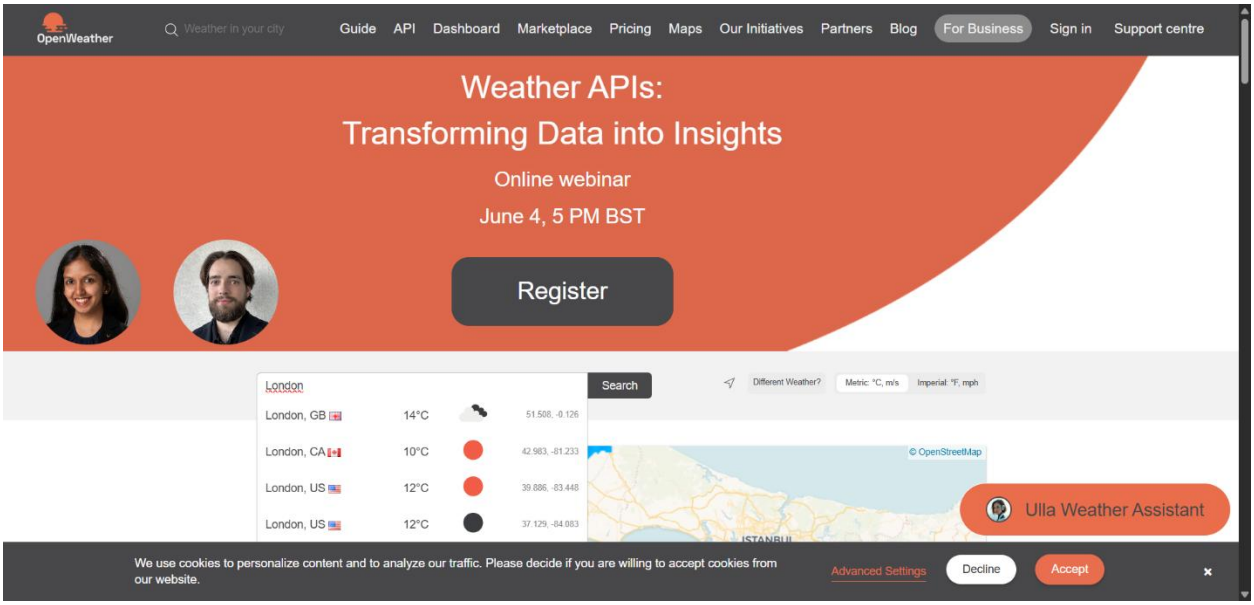
Şekil 21: Arama kutusuna hiçbir şey yazılmadan yapılan sorgunun sonucu; sayfa değişmedi

UI-05 – Büyük-küçük harf duyarlılığı

“london” ve “London” şeklinde iki ayrı arama yapılmış, her iki aramada da aynı sonucun gelip gelmediği kontrol edilmiştir. Sistem büyük-küçük harf farkına duyarlı olmamalıdır.



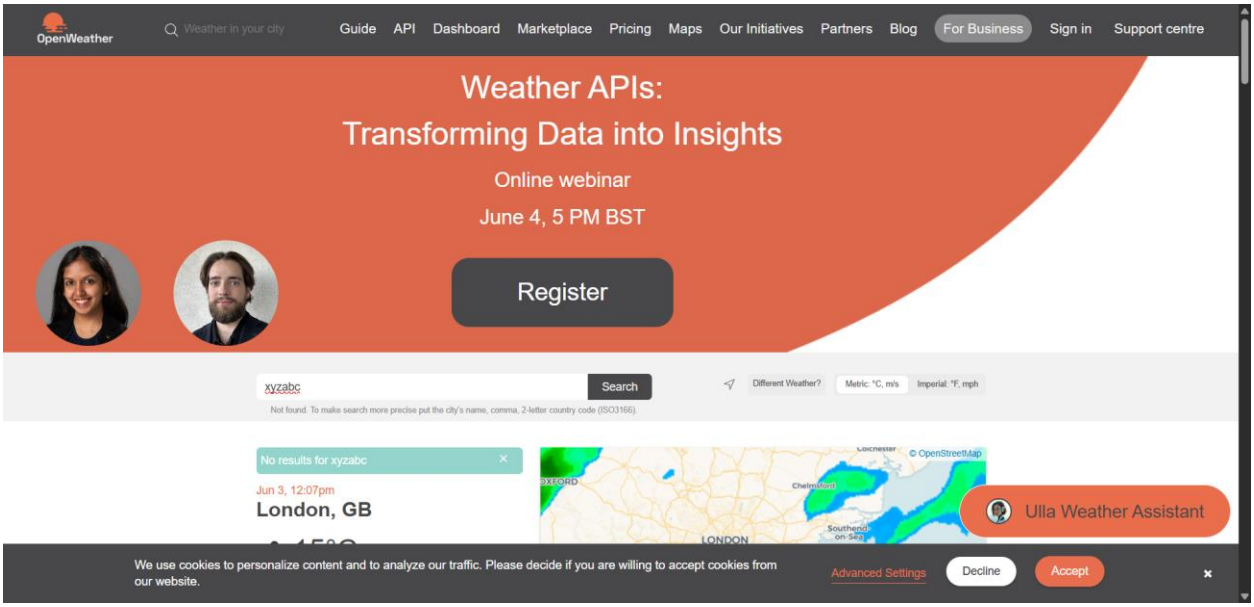
Şekil 22: 'london' araması sonucu



Şekil 23: 'London' araması sonucu

UI-06 – Geçersiz şehir

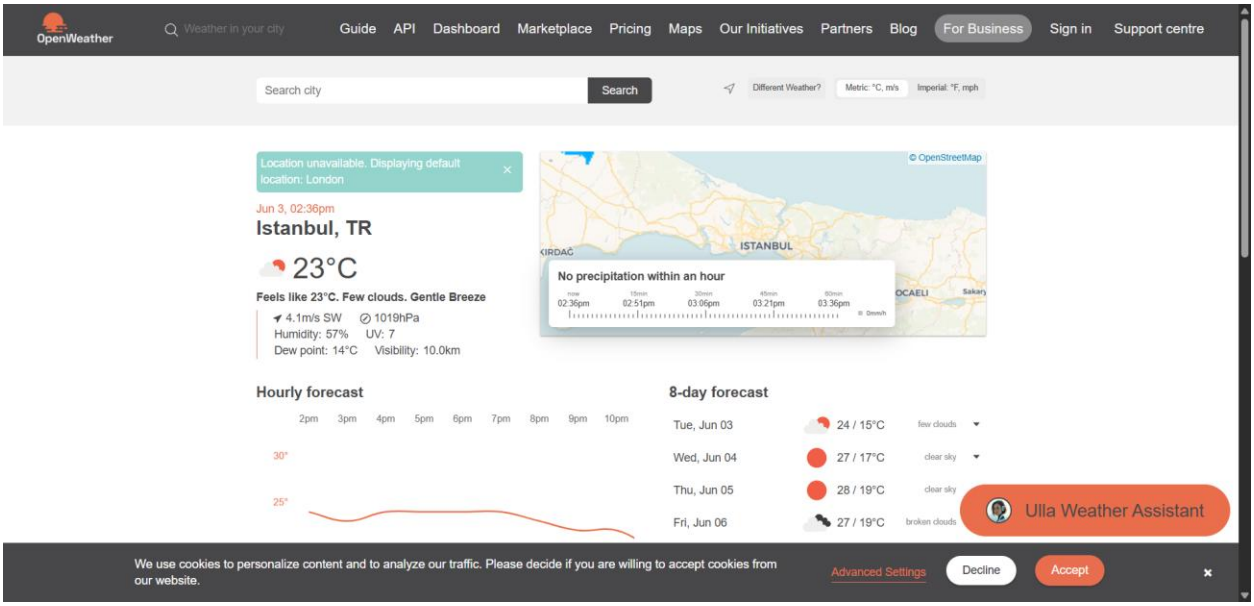
“xyzabc” gibi geçersiz bir şehir arandığında sistemin uygun bir hata mesajı gösterip göstermediği test edilmiştir. Kullanıcıyı yönlendiren bir “City not found” mesajı beklenir.



Şekil 24: “xyzabc” gibi geçersiz bir şehir arandığında çıkan "City not found" mesajı

UI-07 – Dil seçimi testi

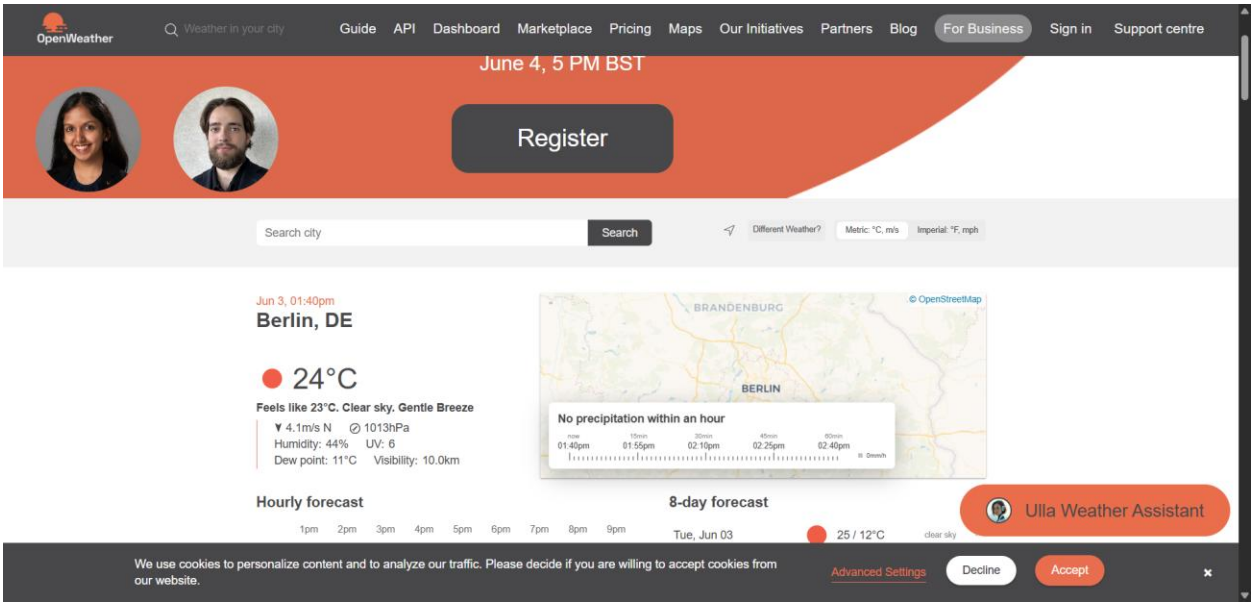
Sayfa Türkçe olarak yüklendiğinde (?lang=tr), hava durumu açıklamalarının Türkçe görünüp görünmediği kontrol edilmiştir. Örneğin “clear sky” yerine “açık” yazmalıdır. Ancak test sonucunda açıklama hâlâ İngilizce kalmıştır, bu nedenle test başarısız sayılmıştır.



Şekil 25: Sayfa Türkçe olarak yüklendiği halde hava durumu açıklamasının İngilizce görünmesi

UI-08 – End-to-End akış

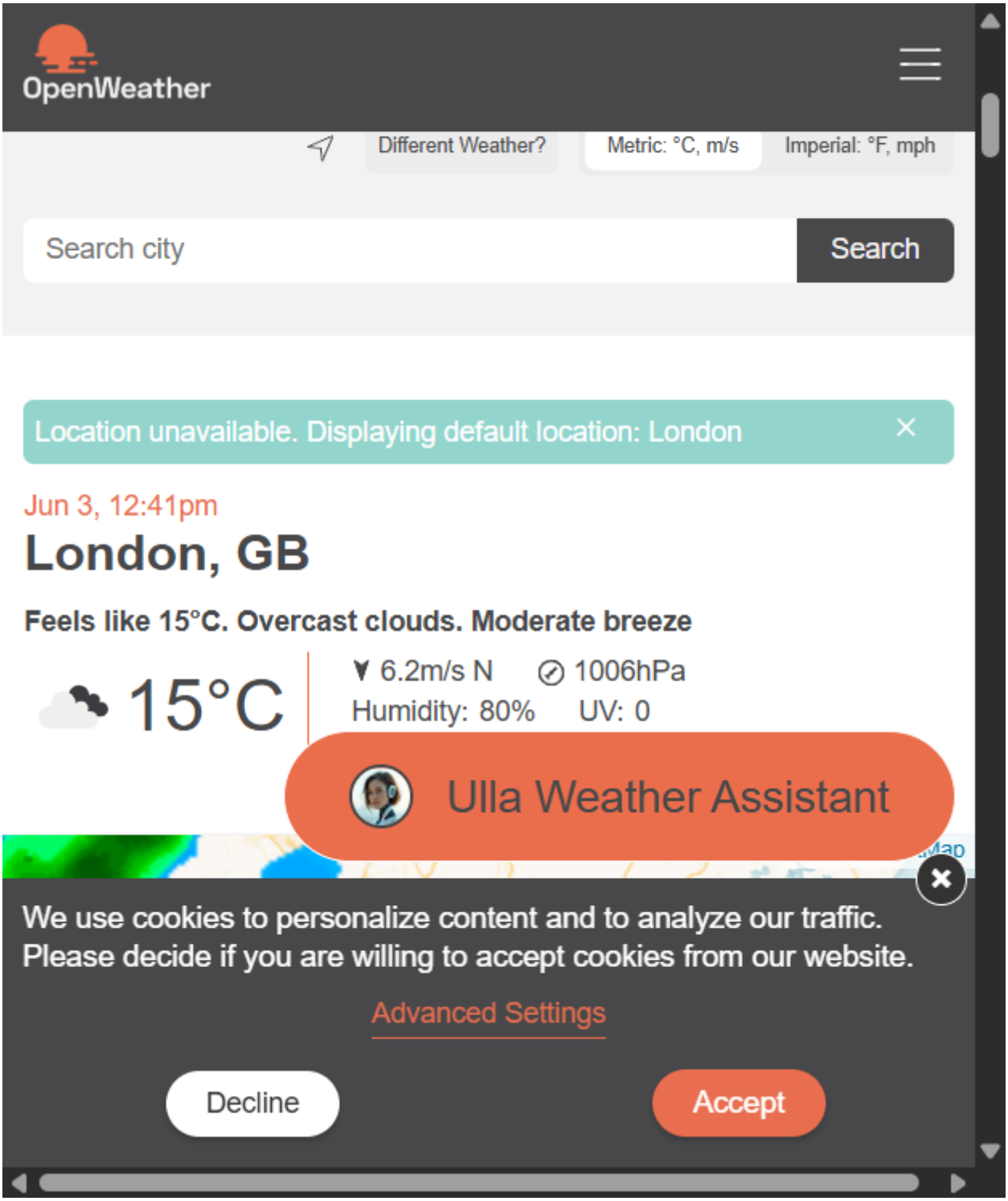
"Berlin" araması yapıldıktan sonra, sonuç seçilmiş ve sayfanın Berlin'e ait sıcaklık bilgilerini düzgün şekilde yükleyip yüklemediği kontrol edilmiştir. Bu bir kullanıcı akışı simülasyonudur.



Şekil 26: Berlin araması sonrası doğru şehir bilgisi ve sıcaklığın görüntülendiği ekran

UI-09 – Mobil görünüm testi

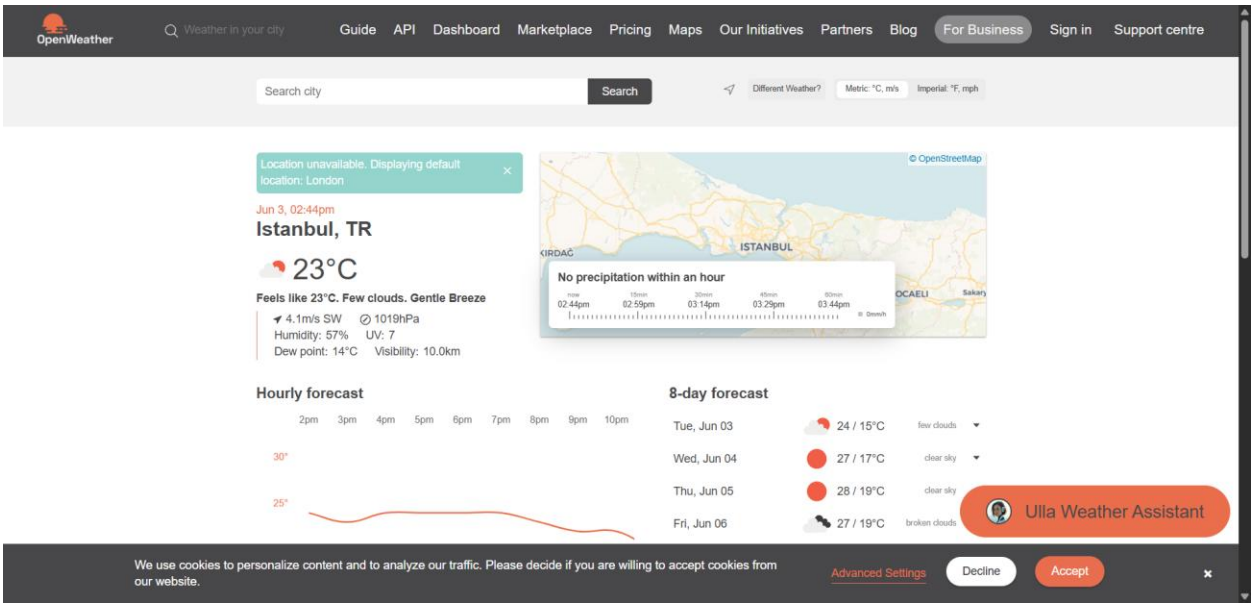
Sayfa 360x740 çözünürlükte (mobil görünüm) açılarak tasarımın bozulup bozulmadığı test edilmiştir. Önemli içeriklerin mobilde de görünür olması beklenir. Sayfa düzgün şekilde yüklenmiş ve test başarılı olmuştur.



Şekil 27: Sayfanın 360px genişliğinde mobil görünümde açılmış hali

UI-10 – UI vs API veri karşılaştırması

UI'daki sıcaklık değeri ile OpenWeatherMap API'sinden dönen sıcaklık değeri karşılaştırılmıştır. Aradaki fark ±1°C'den fazla olmamalıdır. Testte her iki değerin uyumlu olduğu doğrulanmıştır.



Şekil 28: UI ve API sıcaklık karşılaştırması yapılan testten alınan ekran görüntüsü

11. Sonuç

Gerçekleştirilen Selenium tabanlı kullanıcı arayüzü (UI) testleri sonucunda, OpenWeatherMap platformunun temel kullanıcı etkileşimlerine karşı genel olarak başarılı ve tutarlı şekilde yanıt verdiği gözlemlenmiştir. Arama kutusu, butonlar, yönlendirmeler, hata mesajları, dinamik içerikler ve form alanları gibi kritik UI bileşenleri çoğu test senaryosunda beklenen davranışı göstermiştir. Testlerde hem görsel öğelerin yüklenmesi hem de kullanıcı aksiyonlarına verilen sistem yanıtları incelenmiş; bileşenlerin doğru zamanda ve doğru formatta DOM'a eklendiği doğrulanmıştır.

Ancak bazı senaryolarda dinamik içeriklerin geç yüklenmesi nedeniyle testin başarısız olduğu görülmüş, özellikle zamanlamaya duyarlı test adımlarında explicit wait stratejilerinin kullanılmasının önem kazandığı tespit edilmiştir. Ayrıca, responsive tasarım açısından mobil tarayıcılardaki bazı UI öğelerinde hizalama sorunları veya görünürlük

problemleri de raporlanmıştır. Bu durumlar genel kullanıcı deneyimini doğrudan etkileyebileceğinden, görsel tutarlılığın ve etkileşim sürelerinin iyileştirilmesi gerektiği ortaya çıkmıştır.

Genel olarak, sistemin işlevsel ve kullanıcı dostu bir arayüz sunduğu, ancak bazı senaryolarda kullanıcı deneyiminin daha kararlı hale getirilebilmesi için teknik iyileştirmelere ihtiyaç duyulduğu sonucuna varılmıştır.

12. Önerilen / İyileştirmeler

Zamanlama ve Dinamik İçerik Yönetimi:
Arayüzde yer alan dinamik bileşenlerin geç yüklenmesi Selenium testlerinde hata yaratmaktadır. Bunun önüne geçebilmek için UI'nin asenkron veri yüklemelerinde yüklenme tamamlanana kadar kullanıcıya loading göstergesi sunması, testlerde ise explicit wait kullanımının standartlaştırılması önerilmektedir.

Responsive Tasarım İyileştirmeleri:
Mobil ekranlarda bazı bileşenlerin hizalanmaması, butonların dışarı taşması gibi tasarımsal sorunlar tespit edilmiştir. CSS grid/flexbox yapılarıyla bu alanlarda responsive uyum geliştirilmeli, media query kontrolleri gözden geçirilmelidir.

Hata Geri Bildirimlerinin Zenginleştirilmesi:
Geçersiz girişlerde kullanıcıya gösterilen hata mesajları daha açıklayıcı ve yönlendirici hale getirilebilir. Örneğin; "City not found" yerine "The city you entered is not recognized. Please check the spelling or try a nearby location." gibi mesajlar kullanıcı deneyimini iyileştirecektir.

Test Ortamına Özgü DOM Stabilizasyonu:
UI testlerinde bazı elementlerin CSS sınıfları veya ID'leri dinamik olarak değiştiği için test kırılmalarına neden olmuştur. Geliştiricilerin, test ortamları için stabil ve sabit ID'ler veya data-testid attribute'ları tanımlaması önerilmektedir.

Form Doğrulama Geliştirmeleri:
Form bileşenlerine yanlış ya da eksik veri girişi senaryoları test edilmiştir. Bazı durumlarda sistemin bu hataları sessizce geçtiği veya kullanıcıyı uyarmadığı görülmüştür. Alan doğrulamalarının hem görsel hem işlevsel olarak güçlendirilmesi önerilmektedir.