Int. J. Adv. Eng. Pure Sci. 2025, 37(2): <166-187>
DOI: 10.7240/jeps.1604611

TECHNICAL NOTE / TEKNİK NOT

Acil Durum Çağrı Merkezi Uygulamalarında Kullanıcı Memnuniyeti ve Performans Analizi: 112 Örneği

User Satisfaction and Performance Analysis in Emergency Call Center Applications: The Case of 112

Yusuf SELİMDAROĞLU [1], Özgür TONKAL [1]

[1]*Samsun Üniversitesi, Mühendislik ve Doğa Bilimleri Fakültesi, Yazılım Mühendisliği Bölümü, Samsun, Türkiye*

Öz

Bu çalışma, 112 Acil Çağrı Merkezi Uygulamasının işlevselliğini ve kullanıcı memnuniyetini değerlendirmek amacıyla beş test senaryosu tasarlayarak 3056 kişilik bir örneklem üzerinde uygulanmıştır. Test sonuçları ve memnuniyet anketleri istatistiksel yöntemlerle analiz edilerek, uygulamanın performansına dair kapsamlı veriler elde edilmiştir. Çalışmada ayrıca, test senaryolarının kısıtlılıkları ele alınmış ve bu senaryoların daha gerçekçi, kapsamlı ve güvenilir hale getirilmesine yönelik öneriler sunulmuştur. Bulgular, çalışma yılı, cinsiyet ve yaş gibi demografik faktörlerin kullanıcı memnuniyeti üzerindeki etkisini ortaya koymaktadır. Elde edilen veriler, 112 Acil Çağrı uygulamasının iyileştirilmesine yönelik önemli çıktılar sunarak, acil durum müdahale sistemlerinin daha etkin ve verimli çalışmasına katkıda bulunmayı hedeflemektedir.

Anahtar kelimeler: 112 Acil Çağrı Merkezi Uygulamaları, Test Senaryoları, Kullanıcı Memnuniyeti, Acil Durum Müdahale Sistemleri, Yazılım Test Süreçleri

Abstract

This study evaluates the functionality and user satisfaction of the 112 Emergency Call Center Application by designing five test scenarios and applying them to a sample of 3,056 participants. Test results and satisfaction surveys were analyzed using statistical methods, providing comprehensive data on the application's performance. Additionally, the study addresses the limitations of the test scenarios and offers recommendations for making them more realistic, comprehensive, and reliable. The findings reveal the impact of demographic factors such as study year, gender, and age on user satisfaction. The obtained data provide significant insights for improving the 112 Emergency Call Application, contributing to the more effective and efficient operation of emergency response systems.

Keywords: 112 Emergency Call Center Applications, Test Scenarios, User Satisfaction, Emergency Response Systems, Software Testing Processes

I. GİRİŞ

112 acil çağrı uygulamaları, acil durumlarda vatandaşların hızlı ve kolay bir şekilde yardım almasını sağlayan kritik öneme sahip araçlardır [1]. Bu uygulamaların etkin bir şekilde çalışması, insan hayatlarının kurtarılması ve kamu güvenliğinin sağlanması açısından oldukça önemlidir. Bu nedenle, 112 acil çağrı uygulamasının test süreçleri ve memnuniyet değerlendirilmesi, bu uygulamaların başarısının anahtarıdır.

Yazılım test süreçleri, FURPS+ modeli (Functionality - İşlevsellik, Usability - Kullanılabilirlik, Reliability - güvenilirlik, Performance - Performans, Supportability - Desteklenebilirlik) gibi temel kalite kriterlerine dayanarak tasarlanır ve uygulanır [2]. V-Model gibi test odaklı yazılım geliştirme modelleri, testlerin sistem tasarım aşamasından itibaren başlamasını ve geliştirme sürecinin her adımında doğrulama ve doğrulama süreçlerinin uygulanmasını önerir.

112 acil çağrı uygulamalarının test süreçleri, sistemin fonksiyonelliğini, güvenilirliğini ve performansını değerlendirmek amacıyla titizlikle tasarlanmış farklı test türlerinden oluşmaktadır. Birim testleri, sistemin her bir bileşenini ayrı ayrı değerlendirirken; entegrasyon testleri, farklı bileşenlerin birbirleriyle uyumlu çalışıp çalışmadığını analiz eder. Sistem testleri, yazılımın bir bütün olarak ele alınmasını ve performansının değerlendirilmesini sağlarken, kabul testleri, son kullanıcıların beklentilerini karşılayıp karşılamadığını belirlemek için gerçekleştirilir [3].

112 acil çağrı sistemleri için test süreçleri, yalnızca hata tespitine yönelik değil, aynı zamanda kullanıcı deneyiminin iyileştirilmesi ve çağrı merkezlerindeki operasyonel verimliliğin artırılması açısından da önem taşımaktadır. Gerçek kullanıcı senaryolarına dayalı kapsamlı test planları, sistemin gerçek hayattaki acil durum senaryolarında nasıl performans gösterdiğini değerlendirmek için gereklidir.

Sorumlu Yazar: YUSUF SELİMDAROĞLU, **Tel:** 0537 627 51 86, **E-posta:** yusufselimdaroglu@hotmail.com

Gönderilme: 24.12.2024, **Düzenleme:** 24.02.2025, **Kabul:** 05.05.2025

Performans testleri, yüksek çağrı yükü altında sistemin hızını ve yanıt verme süresini test ederken; güvenlik testleri, çağrı merkezi yazılımının siber tehditlere karşı dayanıklılığını ölçmeyi amaçlar [4].

Acil çağrı sistemleri, vatandaşların acil bir durumda en kısa sürede yardım almasını sağlamak amacıyla geliştirilmiş kritik öneme sahip sistemlerdir. Bu sistemler, çağrıları yöneten operatörler aracılığıyla acil hizmet birimlerine yönlendirme yaparak sürecin hızlandırılmasını sağlamaktadır [6]. Acil çağrı merkezleri, coğrafi konum belirleme (GIS), ses analizi, otomatik çağrı yönlendirme ve yapay zeka destekli çağrı yönetim sistemleri gibi teknolojilerle donatılmıştır.

Avrupa'da eCall sistemleri, kaza yapan araçlardan otomatik olarak acil çağrı merkezlerine veri göndererek acil müdahale süreçlerini hızlandırmaktadır [7]. NG112 (Next Generation 112) projeleri, geleneksel sesli çağrıların ötesine geçerek video, SMS ve veri tabanlı çağrı sistemleri ile entegre çalışmayı hedeflemektedir [8]. ABD'de 911 çağrı sistemleri, gelişmiş konum belirleme ve büyük veri analitiği ile çağrıları daha etkin yönlendirebilmek için sürekli olarak geliştirilmektedir.

112 çağrı merkezlerinin işleyişi, çağrının alınması, doğru sınıflandırılması, ilgili birimlere yönlendirilmesi ve geri bildirim mekanizmaları ile sürecin takibini kapsamaktadır [9]. Günümüzde yapay zeka ve makine öğrenimi teknolojileri, önceliklendirilmiş çağrı yönlendirme ve otomatik dil algılama sistemleri ile acil çağrıların yönetilmesine yardımcı olmaktadır. Bu sistemlerin yazılım test süreçleri ile desteklenmesi, çağrı merkezlerinin operasyonel verimliliğini artırarak kritik müdahalelerin zamanında gerçekleştirilmesini sağlamaktadır.

Acil çağrı sistemlerinde kullanılan yazılım ve altyapının güvenilirliği, sistemin etkinliği açısından kritik öneme sahiptir. Yazılım test süreçleri, sistemlerin sorunsuz çalışmasını sağlamak için çeşitli test metodolojilerini içermektedir.

Test süreçlerinde kullanılan başlıca yöntemler şunlardır:
Fonksiyonel Testler – Sistemin belirlenen gereksinimleri karşılayıp karşılamadığını doğrular.
Performans Testleri – Çağrı merkezinin yoğun çağrı altında nasıl çalıştığını değerlendirir.
Yük Testleri – Aşırı çağrı yükü, sunucu çökmesi ve bağlantı kesintisi gibi olağanüstü durumları analiz eder.
Güvenlik Testleri – Veri gizliliği ve yetkisiz erişime karşı sistemin dayanıklılığını ölçer.
Worst-Case Condition Testleri – Acil durum sistemlerinin en stresli koşullarda nasıl çalıştığını belirlemek için uygulanır [10].

NG112 projeleri, acil çağrı sistemlerinin güvenilirliğini artırmak için yeni test süreçleri ve uluslararası standartlar geliştirmektedir [11]. FURPS+ modeli, acil durum yazılımlarında Fonksiyonellik (Functionality), Kullanılabilirlik (Usability), Güvenilirlik (Reliability), Performans (Performance) ve Desteklenebilirlik (Supportability) gibi temel kalite kriterlerini ölçmek için yaygın olarak kullanılmaktadır [12].

Son yıllarda yazılım test otomasyonu, acil çağrı sistemlerinin daha güvenilir hale gelmesi için yaygın olarak kullanılmaktadır. Otomatik test senaryoları, manuel testlere kıyasla daha hızlı ve kapsamlı sonuçlar sunarak, 112 çağrı merkezleri için daha az hata payı ve daha yüksek operasyonel verimlilik sağlamaktadır [13]. CI/CD (Continuous Integration / Continuous Deployment) süreçleri ile otomasyon testleri entegre edilerek sistemin her güncelleme sonrasında test edilmesi sağlanmaktadır.

Acil çağrı sistemlerinde kullanıcı memnuniyeti, sistemlerin başarısını ölçen en önemli faktörlerden biridir. Kullanıcı memnuniyeti, aşağıdaki temel kriterler üzerinden değerlendirilmektedir:
Çağrı Yanıt Süresi – Acil durum çağrılarının ne kadar sürede yanıtlandığını ölçer.
Yönlendirme Doğruluğu – Çağrının en uygun birime ne kadar doğru yönlendirildiğini analiz eder.
Operatör Desteği – Çağrı merkezi personelinin kullanıcıya sağladığı destek hizmetlerini değerlendirir.
Geri Bildirim Süreçleri – Kullanıcıların çağrı sonrası aldıkları hizmetten memnun olup olmadıklarını belirlemek için uygulanır [14].

Kullanıcı memnuniyeti ölçüm yöntemleri arasında anketler, veri analitiği ve performans değerlendirme testleri bulunmaktadır. Anket tabanlı değerlendirme sistemleri, acil çağrı merkezlerinde kullanıcı deneyimini ölçmek için en yaygın kullanılan yöntemlerden biridir. Veri analitiği, çağrı merkezi performansının zaman içindeki değişimini inceleyerek hizmet kalitesini artırmaya yönelik kararların alınmasını sağlamaktadır.

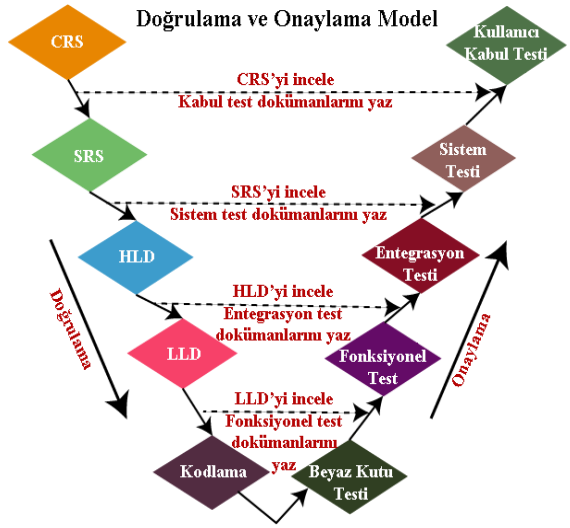
Simülasyon eğitimleri ve saha testleri, operatörlerin ve acil müdahale ekiplerinin gerçekçi senaryolarla eğitilmesini ve hata paylarının azaltılmasını sağlamaktadır. Özellikle yapay zeka destekli çağrı yönlendirme sistemleri, kullanıcıların taleplerine daha hızlı ve doğru şekilde yanıt verilmesini sağlayarak genel memnuniyeti artırmaktadır [15].

Literatürde yapılan çalışmalar, 112 acil çağrı uygulamalarının kullanıcı memnuniyetini artırmak için sürekli geri bildirim mekanizmalarına ve çağrı yönlendirme algoritmalarına yönelik optimizasyon çalışmalarına ihtiyaç duyduğunu göstermektedir [16]. Yapay zeka, büyük veri analizi ve tahmine dayalı modelleme, acil çağrı sistemlerinde kullanıcı deneyimini iyileştirmek için kullanılan yeni teknolojiler arasında yer almaktadır.

Bu çalışma, 112 acil çağrı uygulamalarının test senaryolarını inceleyerek, yazılım geliştirme süreçlerinde hangi test stratejilerinin kullanıldığını değerlendirmekte ve kullanıcı memnuniyeti analizleri ile bu sistemlerin başarısını ölçmeyi amaçlamaktadır. Çalışmanın bulguları, yazılım test süreçlerinin acil durum müdahale sistemlerinin güvenilirliği üzerindeki etkisini anlamaya ve bu sistemlerin daha güvenli ve verimli hale getirilmesine yönelik öneriler geliştirmeye katkı sağlamaktadır.

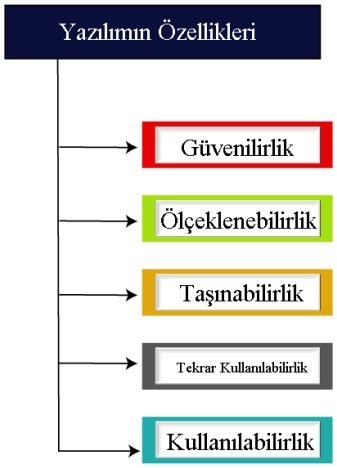
II. YAZILIM TESTİ TEKNİKLERİ

Yazılım testi, bir yazılımın belirli gereksinimlere uygun olarak çalışıp çalışmadığını, doğru ve güvenilir sonuçlar üretmesiyle belirlenen özellikleri yerine getirip getirmediğini değerlendiren bir süreçtir. Yazılım testi süreci, yazılım geliştirme yaşam döngüsünün (SDLC) ayrılmaz bir parçasıdır ve yazılım kalitesini artırmayı, hataları erken tespit etmeyi ve yazılımın kullanıcı gereksinimlerini tam olarak karşılamasını sağlamayı hedefler [17].



Şekil 1. V modelin farklı aşamaları

Bu süreç, sistemin her aşamasında doğrulama ve geçerleme faaliyetlerini içererek yazılımın güvenilirliğini artırır. Şekil 1'de verilen V-Model (V Chart), yazılım geliştirme ve test süreçlerinin birbirine paralel ilerlediğini göstermektedir. Modelin sol tarafı gereksinim analizi, sistem tasarımı ve modül tasarımı gibi geliştirme aşamalarını içerirken, sağ tarafı ise her geliştirme aşamasına karşılık gelen test süreçlerini göstermektedir. Bu yapı, her seviyede testlerin uygulanmasını sağlayarak hataların erken aşamada tespit edilmesine olanak tanır. Böylece, yazılımın daha sağlam ve güvenilir bir şekilde geliştirilmesi sağlanır. V-Model (V Chart) ile yapılandırılan yazılım testi süreci, yazılım kalitesini değerlendirmede kullanılan FURPS+ modelinin temel bileşenleriyle doğrudan ilişkilidir. FURPS+ modeli; İşlevsellik (Functionality), Kullanılabilirlik (Usability), Güvenilirlik (Reliability), Performans (Performance) ve Desteklenebilirlik (Supportability) gibi yazılım kalite kriterlerini içermektedir [12]. V-Model'in sağ tarafında yer alan test süreçleri, FURPS+ kriterlerinin her birini sistematik olarak doğrulamak için uygulanmaktadır. Şekil 2' de bir yazılımın sahip olması gereken özellikler verilmiştir. 112 Acil Çağrı Merkezi Uygulaması-Yazılımı bu kapsamda geliştirilmiş olup FURPS+ kriterlerine göre değerlendirmelerde bulunulmuştur.



Şekil 2. Yazılımın özellikleri

Fonksiyonellik (Functionality), yazılımın istenilen gereksinimleri karşılayıp karşılamadığını test eden bir kriterdir. Fonksiyonel testler, yazılımın kullanıcıya sunduğu işlevlerin doğru bir şekilde çalışıp çalışmadığını değerlendirir. 112 acil çağrı merkezi yazılımında, fonksiyonellik testleri, yazılımın çağrı yönlendirme, veri işleme, hata yönetimi gibi temel işlevlerinin düzgün çalışıp çalışmadığını kontrol eder. Bu tür testler, yazılımın beklenen özellikleri yerine getirip getirmediğini tespit eder [1].

Kullanılabilirlik (Usability), yazılımın ne kadar kullanıcı dostu olduğuna odaklanır. Kullanıcıların yazılımı öğrenme süresi, kullanım kolaylığı ve hatalı işlem yapma olasılıkları, kullanılabilirlik testlerinin hedeflediği alanlardır. 112 acil çağrı merkezi yazılımında, kullanılabilirlik testleri, operatörlerin sistemle etkileşimini değerlendirmek için yapılır. Kullanıcı arayüzünün ne kadar sezgisel olduğu, operatörlerin çağrıları hızlı ve doğru bir şekilde yönetebilmesi için önemli bir faktördür. Kullanıcı memnuniyetini artırmak için sistemin kullanım kolaylığı test edilir [4].

Güvenilirlik (Reliability), yazılımın hata yapmadan ne kadar süreyle çalıştığını ve sistemin kararlılığını ölçen bir kriterdir. Yazılımın hatasız bir şekilde çalışması, acil durumlarda hayat kurtarıcı olabilir. 112 acil çağrı merkezi yazılımında, güvenilirlik testleri, yazılımın uzun süreli kullanımda hata vermeden çalışıp çalışmadığını belirler. Yazılımın sistem arızalarına karşı dayanıklı olup olmadığı, acil durumlarda

kesintisiz hizmet sunabilmesi için kritik bir faktördür. Güvenilirlik testleri, yazılımın başarısızlık durumlarında ne kadar hızlı toparlanabileceğini de değerlendirir [11].

Performans (Performance), yazılımın hızını ve yanıt verme süresini ölçen bir kriterdir. Yüksek performans, yazılımın hızlı çalışmasını ve yüksek veri işlem kapasitesine sahip olmasını sağlar. 112 acil çağrı merkezi yazılımı gibi yüksek trafik altında çalışan sistemlerde, performans testleri, yazılımın yoğun çağrı trafiği altında nasıl çalıştığını değerlendirir. Bu testler, yazılımın yanıt sürelerini, veri işleme hızını ve kullanıcı taleplerine ne kadar hızlı cevap verdiğini ölçer. Performans testleri, yazılımın acil durumlarda hızla yanıt verebilmesi için gereklidir [7].

Desteklenebilirlik (Supportability), yazılımın bakımının ne kadar kolay yapıldığını ve yazılımın uzun vadeli sürdürülebilirliğini değerlendirir. Yazılım geliştirme sürecinde, yazılımın bakımını ve güncellemelerini kolaylaştırmak için testler yapılmalıdır. 112 acil çağrı merkezi yazılımında, desteklenebilirlik testleri, yazılımın hatalarını ne kadar hızlı bir şekilde düzeltebileceğimizi, güncelleme ve entegrasyon süreçlerinin ne kadar kolay olduğunu ölçer. Ayrıca, yazılımın gelecekteki sürümleriyle uyumlu olup olmadığını da test eder [3].

FURPS+ modelinin artı işareti, yazılımın taşınabilirlik, güvenlik, uyumluluk gibi ek kriterleri de içerdiğini gösterir. Bu ek kriterler, yazılımın çevresel faktörlere ne kadar uyum sağlayabileceğini, diğer sistemlerle ne kadar entegre olabileceğini ve sistemin güvenliğini nasıl koruyacağını değerlendirir. Özellikle 112 acil çağrı merkezlerinde, yazılımın diğer sistemlerle uyumlu çalışabilmesi, güvenliğinin sağlanması ve taşınabilir olması önemlidir. Bu tür testler, yazılımın daha geniş bir uygulama alanına sahip olmasını ve gelişen teknolojiye uyum sağlamasını garanti eder.

Sonuç olarak, FURPS+ modeli, yazılım test süreçlerinde geniş bir perspektif sunarak, yazılımın yalnızca fonksiyonel gereksinimlere değil, aynı zamanda kullanıcı memnuniyetine, sistemin güvenliğine, performansına ve uzun vadeli sürdürülebilirliğine de odaklanılmasını sağlar. Bu model, yazılımın her yönünü test ederek, daha kaliteli, güvenli ve verimli bir ürün ortaya koymayı hedeflemektedir.

2.1. Yazılım Geliştirme Döngüsü

Yazılım geliştirme döngüsü, bir yazılımın ilk tasarım aşamasından son kullanıcıya sunulmasına kadar geçen süreci kapsar. Şekil 1 ve Şekil 3' de verilen bu döngüde yazılımın işlevselliği, performansı, güvenliği ve kullanıcı deneyimi gibi birçok faktör sürekli olarak test edilir. Yazılım geliştirme süreçlerinde testlerin nasıl entegre edileceği, yazılımın kalitesini doğrudan etkiler. Test süreçlerinin yazılım geliştirme döngüsüne doğru şekilde dahil edilmesi, yazılımın güvenilirliğini ve verimliliğini artırır.

Yazılım geliştirme döngüsü genellikle birkaç temel aşamadan oluşur: gereksinim toplama, tasarım, geliştirme, test ve bakım. Bu aşamalarda her biri birbirini takip eden adımlar olsa da, yazılımın başarılı bir şekilde tamamlanabilmesi için her aşamanın doğru bir şekilde entegre edilmesi gerekmektedir. Testler, yazılım geliştirme sürecinin her aşamasında yapılmalı ve yazılımın her yönü kapsamlı bir şekilde değerlendirilmelidir. V-Model ve Agile gibi yazılım geliştirme modelleri, test süreçlerinin yazılım geliştirme sürecine entegrasyonuna yönelik sistematik bir çerçeve sunarak, yazılım kalitesinin artırılmasına ve hata tespitinin erken aşamalarda gerçekleştirilmesine olanak tanımaktadır.

V-Modeli, yazılım geliştirme sürecinde test süreçlerinin entegrasyonu için yaygın olarak kullanılan bir yaklaşımdır. Bu model, geliştirme sürecini iki aşamaya ayırır: sol taraf geliştirme aşamalarını (gereksinim toplama, tasarım) ve sağ taraf ise test aşamalarını (birim testi, entegrasyon testi, sistem testi ve kabul testi) içerir. V-Modeli, yazılımın her aşaması için karşılık gelen bir test aşaması belirler. Yazılım geliştirmeye başlandığında, ilk olarak gereksinimlerin belirlenmesi ardından tasarım aşamasına geçilir. Bu aşamalardan her biri, karşılık gelen testlerle birlikte planlanır. Örneğin, yazılımın her bir bileşeni için geliştirme yapılırken, buna uygun birim testleri de paralel olarak yürütülür. Sistem testi ise yazılımın tüm bileşenlerinin birlikte nasıl çalıştığını değerlendirir. Bu şekilde, test süreçleri yazılım geliştirme döngüsünün her aşamasında doğrudan yer alır [15].

Şekil 3. Yazılım geliştirme döngüsünün aşamaları

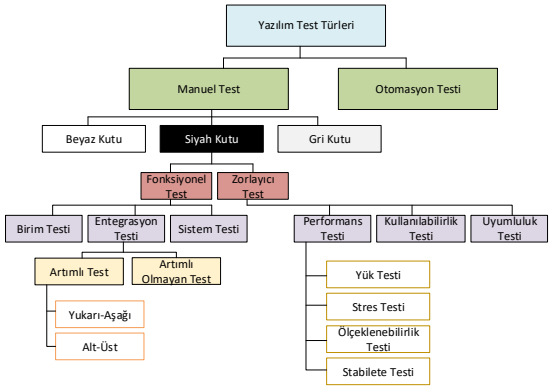
Agile yazılım geliştirme, yazılımın küçük, yönetilebilir parçalar halinde geliştirilmesini ve her parçanın test edilmesini sağlar. Agile, yazılım geliştirme sürecini kısa döngüler (sprintler) halinde organize eder ve her sprint sonunda yazılımın geliştirilmiş bir sürümü test edilir. Agile geliştirme sürecinde, yazılımın her yeni sürümünde fonksiyonellik, güvenilirlik ve performans gibi kriterler sürekli olarak test edilir. Agile, yazılımın hızla değişen gereksinimlere adapte olabilmesini sağlar. Testler ise her iterasyonda yapılır, bu sayede yazılım geliştirme sürecine entegre edilmiş sürekli geri bildirim döngüsü oluşturulmuş olur. Bu, yazılımın kalitesini artırmak için oldukça etkili bir yöntemdir çünkü kullanıcı geri bildirimleri doğrultusunda yazılım sürekli olarak iyileştirilir [16].

Her iki modelde de testler, yazılım geliştirme sürecinin ayrılmaz bir parçasıdır. V-Modeli, yazılım geliştirme aşamalarıyla paralel olarak testlerin yapılmasını sağlarken, Agile modeli daha esnek bir yaklaşım sunarak, her sprint sonunda testlerin tekrar yapılmasını ve yazılımın sürekli olarak iyileştirilmesini sağlar.

Yazılım geliştirme döngüsüne test süreçlerinin entegre edilmesi, yazılımın her yönünün güvenli, kullanıcı dostu ve hatasız olmasını sağlar. Özellikle 112 acil çağrı merkezi gibi kritik öneme sahip sistemlerde, bu süreçlerin titizlikle yönetilmesi gerekir. Her iki modelde, yazılım geliştirme sürecinde testlerin doğru zamanda ve doğru şekilde yapılmasını sağlayarak, yazılımın son kullanıcıya en yüksek kalitede sunulmasına katkı sağlar.

2.2. Yazılım Testi Türleri

Yazılım testi, bir yazılımın kalitesini ve işlevselliğini değerlendirmek için kullanılan önemli bir süreçtir. Şekil 4' de gösterilen test türleri, yazılımın farklı yönlerini test etmek için kullanılır ve her birinin kendine özgü avantajları ve dezavantajları vardır.



Şekil 4. Yazılım testi türleri

Bazı önemli test türleri şunlardır:

- **Manuel Test:** Bu test türünde, testler insanlar tarafından manuel olarak gerçekleştirilir. Beyaz kutu testi, kara kutu testi ve gri kutu testi gibi farklı koşullarda bu test uygulanabilir.
- **Otomatik Test:** Bu test türünde, testler otomatik araçlar tarafından gerçekleştirilir. Otomatik test, manuel testten daha hızlı ve daha verimli olabilir, ancak daha fazla önceden planlama ve kurulum gerektirir.
- **Duman Testi:** Bu test türünde, yazılımın temel işlevleri düzgün çalışıp çalışmadığını hızlı bir şekilde kontrol etmeyi amaçlar. Bu test ile bir sonraki geliştirme aşamasına hazır olunup olunmadığı test edilmiş olur.
- **Akıl Sağlığı Testi:** Bu test türünde, yazılımın beklenen şekilde davrandığını ve beklenmedik hatalar içermediğini doğrulamak için temel işlevler test edilir.
- **Gerileme Testi:** Bu test türünde, yeni bir değişikliğin var olan işlevselliği bozmadığından emin olmak için eski testler tekrar çalıştırılır.
- **Kullanıcı Kabul Testi:** Bu test türünde, yazılımın nihai kullanıcılar tarafından test edilmesi ve kabul kriterlerini karşılayıp karşılamadığını belirlemesi sağlanır.
- **Güvenlik Testi:** Bu test türünde, yazılımdaki güvenlik açıkları ve zafiyetler aranır.
- **Küreselleşme Testi:** Bu test türünde, yazılımın farklı dilleri, kültürleri ve bölgeleri destekleyip desteklemediği test edilir.
- **FURPS+ Testi:** Yazılımın işlevsellik, kullanılabilirlik, güvenilirlik, performans ve desteklene bilirlik gibi temel kalite kriterlerini değerlendiren bir test türüdür. Ek olarak, **güvenlik, ölçeklenebilirlik, uyumluluk ve sürdürülebilirlik** gibi ek faktörler de "+" kapsamında ele alınır.

Yazılım testi için hangi test türlerinin kullanılacağı, projenin boyutuna, karmaşıklığına ve risklerine bağlıdır. En iyi sonuçlar için, genellikle birden fazla test türü kullanılır [14].

2.3. Yazılım Test Süreçleri

Yazılım test süreci, bir yazılımın ne kadar iyi çalıştığını ve gereksinimleri karşılayıp karşılamadığını değerlendirmek için kullanılan bir dizi adımdır. Bu süreç, yazılımın kalitesini ve güvenilirliğini artırmak, hataları erken aşamada bulmak ve müşteri memnuniyetini sağlamak için önemlidir [15]. Testin hangi amaç ile yapıldığı belirlemelidir. Birim test midir, geliştirme testi midir, kabul testi midir, tip testi midir? Ona göre aşağıdaki adımlar belirlenebilir.

Yazılım test süreci genel olarak şu adımları içerir:

1. Planlama: Test planı oluşturma aşaması, bir projenin başarılı bir şekilde test edilmesi için kritik önem taşır. Bu aşamada, test edilecek yazılımın veya sistemin kapsamı ve hedefleri net bir şekilde tanımlanır. Amaca göre gerekli yöntemler (okuyarak, manuel koşulu, otomatik koşulu), test derinliği (beyaz, gri, kara kutu testleri), test araçları, ekipman ve personel gibi kaynaklar belirlenir ve tahsis edilir. Son olarak, testlerin gerçekleştirileceği ortam kurulur ve hazır hale getirilir. Bu aşamadaki tüm planlama ve hazırlık çalışmaları, testlerin sorunsuz ve verimli bir şekilde yürütülmesine zemin hazırlar.

2. Tasarım: Test senaryoları tasarlama aşaması, test planında belirlenen hedeflere ulaşmak (V Chart'a göre daha tasarım aşamasında bunlar belirlenmiş olmalıdır) için gerekli adımları ve kontrolleri detaylandırır. Bu aşamada, her bir test senaryosu için önkoşullar, adımlar, beklenen sonuçlar ve değerlendirme kriterleri tanımlanır. Test verileri, test senaryolarına uygun şekilde hazırlanır ve organize edilir. Kullanıcı dostu test araçları ve metodolojileri seçilerek testlerin

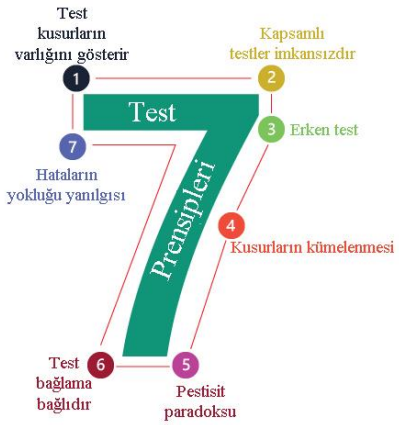
otomatikleştirilmesi ve verimliliği artırılır. Bu aşamadaki çalışmalar, testlerin kapsamlı ve doğru bir şekilde yürütülmesini sağlar.

3. Uygulama: Test senaryolarının yürütülmesi aşaması**,** planlanmış test senaryolarının sırayla ve eksiksiz bir şekilde gerçekleştirildiği aşamadır. Bu aşamada, test uzmanları her bir test senaryosunu manuel veya otomatik olarak çalıştırır. Beklenen sonuçlarla gerçek sonuçlar karşılaştırılır ve herhangi bir tutarsızlık veya hata durumunda detaylı raporlar hazırlanır. Raporlarda, hataların tanımı, tekrar gözlenmesini sağlayacak adımları, ekran görüntüleri ve diğer ilgili bilgiler yer alır. Amaca göre düzeltici faaliyet için sonuç dokümanı hazırlanabilir, istisnası birim testidir. Bu sayede, geliştiriciler hataları analiz edebilir ve gerekli düzeltmeleri yapabilir.

4. Doğrulama: Hata düzeltme ve doğrulama aşaması**,** test sürecinin son aşamasıdır. Bu aşamada, test sırasında raporlanan hatalar geliştiriciler tarafından analiz edilir ve gerekli düzeltmeler yapılır. Düzeltmelerin ardından, test senaryoları tekrar yürütülerek hataların düzeltildiği doğrulanır. Test senaryoları tavsiye olarak tamamen, kesintisiz olarak yürütülmelidir. Gerileme testinin raporlama öncesinde uygulanması, sürecin güvenilirliğini etkileyebilir ve dikkatle ele alınmalıdır. Son olarak, yazılımın tüm kabul kriterlerini karşıladığından ve üretim ortamına hazır olduğundan emin olunur. Bu aşamada yapılan titiz çalışmalar, yazılımın kalitesini ve güvenilirliğini artırır. Tespit edilen hatalar düzeltilir.

5. Raporlama: Test sonuçlarının raporlanması aşaması, test sürecinin tamamlanmasından sonra elde edilen bulguları ve önerileri raporlama aşamasıdır. Bu aşamada, test planı, test senaryoları, yürütülen testler, karşılaşılan hatalar, yapılan düzeltmeler ve elde edilen tüm veriler kapsamlı bir şekilde raporlanır. Raporlar, ilgili paydaşlarla paylaşılır ve yazılımın genel durumu, performansı ve kalitesi hakkında bilgi verilir. Ayrıca, gelecekteki geliştirmeler için önerilerde bulunulur. Bu sayede, paydaşlar yazılımın durumunu şeffaf bir şekilde görebilir ve gerekli adımları alabilirler.

Yazılım test süreçleri, yazılımın kalitesini sağlamak ve hatalarını erken aşamalarda tespit etmek amacıyla yapılan bir dizi değerlendirme aşamasıdır. Bu test süreçleri genellikle farklı seviyelerde yapılır ve her seviyede farklı hedeflere ulaşılmaya çalışılır. Şekil 5'te, yazılım test süreçlerinde temel kabul edilen yedi test ilkesi şematize edilmiştir: (1) Test kusurların varlığını gösterir, (2) Kapsamlı testler imkânsızdır, (3) Erken test, (4) Kusurların kümelenmesi, (5) Pestisit paradoksu, (6) Test bağlama bağlıdır ve (7) Hataların yokluğu yanılgısı. Bu prensipler, test süreçlerinin etkinliğini artırmak ve yazılım kalitesini iyileştirmek için rehber niteliği taşımaktadır. Yazılım test süreçleri, bu ilkeler doğrultusunda dört temel seviyede yürütülmektedir: Birim Testi, Entegrasyon Testi, Sistem Testi ve Kabul Testi. Her bir test seviyesi, yazılımın belirli bir yönünü değerlendirerek, geliştirme sürecinin her aşamasında kaliteyi güvence altına almak için kritik bir rol oynamaktadır.



Şekil 5. Yedi farklı test ilkesi

Birim testi (Unit Testing), yazılım geliştirme sürecinin en temel test seviyesidir ve genellikle yazılımın en küçük bileşenlerini, yani fonksiyonları, metodları veya sınıfları test etmek için kullanılır. Birim testleri, yazılımın her bir bileşeninin doğru şekilde çalışıp çalışmadığını kontrol eder. Yazılımın her modülü, diğer modüllerden bağımsız olarak test edilmelidir. Bu testlerin amacı, her bir bileşenin belirlenen işlevleri doğru şekilde yerine getirip getirmediğini doğrulamaktır. Örneğin, 112 acil çağrı merkezi yazılımında, birim testleri, çağrı yönlendirme fonksiyonunun doğru çalışıp çalışmadığını test edebilir. Bu testler, yazılım geliştirme sürecinin başında yapılmalı ve yazılımın her bileşeninin hatasız olmasını sağlamalıdır [15].

Entegrasyon testi (Integration Testing), yazılımın farklı bileşenlerinin birbirleriyle uyum içinde çalışıp çalışmadığını kontrol eder. Birim testlerinde bağımsız olarak test edilen bileşenler, entegrasyon testinde bir araya getirilir ve sistemin tüm parçalarının birbirleriyle nasıl etkileşime girdiği değerlendirilir. Bu test türü, özellikle yazılımın birden fazla modül veya servis içerdiği durumlarda önemlidir. 112 acil çağrı merkezi yazılımında, entegrasyon testleri, çağrı alma, yönlendirme ve veritabanı işlemleri gibi farklı modüllerin uyumlu çalışıp çalışmadığını kontrol eder. Bu seviyede yapılan testler, sistemin bileşenleri arasındaki veri akışının doğruluğunu ve uyumluluğunu değerlendirir [16].

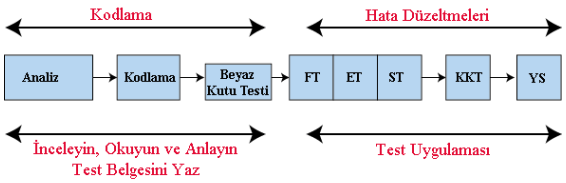
Sistem testi (System Testing), yazılımın tüm bileşenlerinin bir arada çalışıp çalışmadığını ve tüm sistemin genel olarak belirtilen gereksinimleri karşılayıp karşılamadığını test eder. Sistem testi, yazılımın entegrasyon aşamasından sonra yapılır ve yazılımın toplam işlevselliğini değerlendirir. Bu test, yazılımın her yönünü kapsar ve genellikle yazılımın tüm modüllerinin bir arada çalıştığı senaryolar üzerinden gerçekleştirilir. 112 acil çağrı merkezi

yazılımında, sistem testleri, çağrı alma, yönlendirme, veri işleme ve sonuç raporlama gibi tüm süreçlerin doğru bir şekilde çalışıp çalışmadığını kontrol eder. Sistem testi, yazılımın kullanıcı senaryoları altında nasıl çalıştığını ve tüm işlevlerin birlikte nasıl performans gösterdiğini ölçer [3].

Kabul testi (Acceptance Testing), yazılımın son kullanıcı gereksinimlerini karşılayıp karşılamadığını değerlendirmek için yapılan son test aşamasıdır. Bu test, yazılımın müşteri veya kullanıcılar tarafından kabul edilmeden önce yapılır ve genellikle gerçek kullanıcıların test senaryoları üzerinde gerçekleştirilir. Kabul testinin amacı, yazılımın belirtilen işlevleri doğru şekilde yerine getirip getirmediğini, kullanıcı gereksinimlerini karşılayıp karşılamadığını tespit etmektir. 112 acil çağrı merkezi yazılımında, kabul testleri, yazılımın operatörler tarafından nasıl kullanılacağı, çağrıların doğru bir şekilde yönlendirilip yönlendirilmediği gibi unsurlar üzerinden yapılır. Kullanıcı geri bildirimlerine dayalı olarak yapılan bu testler, yazılımın son kullanıcı tarafından kabul edilip edilmeyeceğini belirler [17].

Her bir test seviyesi, yazılımın farklı bir yönünü hedef alır ve yazılımın her bileşeninin veya tüm sistemin hatasız çalışmasını sağlamaya yardımcı olur. Bu testler, yazılımın kalite seviyesini yükseltir, hataları erkenden tespit eder ve yazılımın son kullanıcıya en yüksek kalitede sunulmasını sağlar. 112 acil çağrı merkezi gibi kritik öneme sahip yazılımlarda, her seviyedeki testlerin etkin bir şekilde yapılması, yazılımın güvenli, hızlı ve hatasız çalışmasını garanti eder.

Test senaryoları, yazılımın doğru çalışıp çalışmadığını değerlendirmek için oluşturulan önemli araçlardır. Bu senaryolar, yazılımın işlevselliğini, güvenliğini, performansını ve kullanıcı deneyimini test etmek için belirli adımları içerir. Test senaryoları, yazılım geliştirme sürecinde yazılımın beklenen gereksinimlere göre nasıl çalıştığını doğrulamak amacıyla yapılır. Senaryo geliştirme sürecinde ilk adım, yazılımın tüm gereksinimlerini doğru şekilde anlamak ve bu gereksinimlere dayanarak test senaryoları oluşturmak olmalıdır. Her test senaryosu, yazılımın belirli bir fonksiyonunu ya da özelliğini test etmek için tasarlanmalıdır. Özellikle, 112 acil çağrı merkezi yazılımı gibi kritik sistemlerde, test senaryolarının gerçekçi olması büyük önem taşır. Bu tür yazılımlarda, senaryolar yalnızca normal kullanım senaryolarını değil, aynı zamanda olağanüstü durumları ve hatalı girişleri de kapsamalıdır. Örneğin, bir test senaryosu, bir acil durumda, yazılımın çağrıyı doğru bir şekilde alıp almadığını, veritabanı işlemlerinin doğru yapılıp yapılmadığını ve çağrının doğru birime yönlendirilip yönlendirilmediğini kontrol edebilir [1].

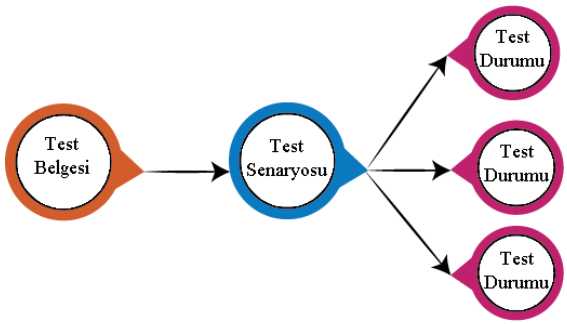


Şekil 6. Test yürütme süreci

Test senaryoları geliştirilirken, kapsamlı olmalarına özen gösterilmelidir. Yazılımın tüm yönlerinin test edilmesi, olası hataların erken aşamalarda tespit edilmesine olanak sağlar. Bu, yazılımın tüm işlevlerinin doğru şekilde çalıştığını ve kullanıcıların yazılımı en verimli şekilde kullanabildiğini garanti altına alır. Ayrıca, senaryoların oluşturulmasında yazılımın gereksinimlerinin tam olarak anlaşılması çok önemlidir. Her test senaryosunun, yazılımın hangi özelliğini test ettiği, bu testin ne amaçla yapıldığı ve beklenen sonuçların ne olduğu açık bir şekilde belirtilmelidir. Gereksinimlerin doğru bir şekilde belirlenmesi, testin ne kadar verimli olacağını doğrudan etkiler. 112 acil çağrı merkezi yazılımı gibi bir uygulamada, test senaryolarının her zaman kullanıcı geri bildirimlerine dayalı olarak oluşturulması ve kullanıcıların acil durumlarda nasıl etkileşimde bulunacağı göz önünde bulundurulmalıdır [3]. Şekil 6' da Test yürütme süreci gösterilmiştir. Testlerin başarılı olabilmesi için, her senaryo belirli adımlarla test edilen işlevi izlemeli ve yazılımın tüm işlevsel gereksinimlerini kapsamalıdır. Bu şekilde, yazılımın tüm bileşenlerinin uyum içinde çalışıp çalışmadığı anlaşılabilir.

Son olarak, test senaryolarının oluşturulmasında dikkat edilmesi gereken bir diğer önemli husus, senaryoların gerçekçi ve tekrarlanabilir olmasıdır. Yazılımın her güncel sürümünde testlerin doğru bir şekilde yapılabilmesi için senaryoların doğru ve anlaşılır bir şekilde yazılması gerekir. Testlerin her birinin belirli bir hedefi olmalı ve test sonuçları yazılımın gereksinimlerine uygun olarak doğru bir şekilde değerlendirilmelidir. Bu sürecin sonunda elde edilen test senaryoları, yazılımın eksiksiz ve hatasız bir şekilde çalışıp çalışmadığını değerlendirmek için çok değerli bir araç olacaktır [15].

Yazılım test süreçlerinde kullanılan teknikler, yazılımın her yönünü detaylı bir şekilde değerlendirmeyi sağlar. Bu teknikler, testin kapsamını genişletmek, hataları erken tespit etmek ve yazılımın kaliteli olmasını sağlamak için kullanılır. Test senaryoları için yaygın olarak kullanılan tekniklerden bazıları black-box testi, white-box testi, grey-box testi ve eşleme testi gibi yöntemlerdir. Bu tekniklerin her biri, yazılımın farklı yönlerini ve işlevlerini test etme amacı taşır ve her birinin avantajları bulunmaktadır. Şekil 7' de Test senaryo süreci gösterilmiştir.



Şekil 7. Test senaryosu süreci

Black-box testi, yazılımın iç yapısına dair hiçbir bilgiye sahip olmadan, yalnızca dışa açık işlevlerin doğruluğunu test etmeye odaklanır. Bu teknik, yazılımın kullanıcı tarafından görülebilen işlevlerinin doğru çalışıp çalışmadığını kontrol eder. Örneğin, 112 acil çağrı merkezi yazılımında, black-box testi çağrı yönlendirme, veri giriş çıkışları ve kullanıcı etkileşimlerinin doğruluğunu kontrol etmek için kullanılır. Bu teknik, yazılımın dışa dönük işlevselliğini test ederken, geliştiricinin yazılımın iç yapısına dair bilgi sahibi olmasına gerek bırakmaz ve özellikle kullanıcı deneyiminin değerlendirilmesi açısından büyük avantaj sağlar [1].

White-box testi, yazılımın iç yapısına dair ayrıntılı bilgiye sahip olarak yapılan testtir. Bu testte, yazılımın kaynak kodu, algoritmalar ve mantıksal akış detaylı bir şekilde incelenir. White-box testi, yazılımın her satırını ve fonksiyonunu gözden geçirerek, yazılımdaki potansiyel hataları bulmaya çalışır. 112 acil çağrı merkezi yazılımında, white-box testi, yazılımın çağrı yönlendirme algoritmalarının doğru çalışıp çalışmadığını, veri işleme süreçlerinin düzgün yapılıp yapılmadığını kontrol etmek için kullanılabilir. Bu teknik, yazılımın derinlemesine analiz edilmesini sağlar ve küçük hataların bile tespit edilmesine yardımcı olur [15].

Bir diğer teknik olan grey-box testi, black-box ve white-box testlerinin birleşimidir. Bu teknik, yazılımın iç yapısı hakkında sınırlı bilgiye sahip olmayı gerektirir ve hem dış işlevler hem de iç mantık üzerinde testler yapar. Grey-box testi, yazılımın hem işlevsel gereksinimlerini hem de iç süreçlerini göz önünde bulundurarak daha kapsamlı testler yapılmasını sağlar. 112 acil çağrı merkezi yazılımında, grey-box testi, kullanıcı arayüzünü test ederken aynı zamanda yazılımın iç veri akışını da gözden geçirebilir. Bu, yazılımın her iki yönünü birleştirerek kapsamlı bir test süreci sunar [16].

Son olarak, eşleme testi (equivalence partitioning), test edilen yazılımın giriş değerlerinin belirli sınıflara ayrılarak her bir sınıftan bir değer seçilmesi esasına dayanır. Bu test, her bir girişin doğru şekilde işlendiği ve yazılımın bu farklı girişlere uygun tepkileri verdiği durumları kapsar. 112 acil çağrı merkezi yazılımında, eşleme testi, farklı acil durum türleri (yangın, trafik kazası vb.) ile yazılımın doğru tepki verip vermediğini test edebilir. Bu yöntem, gereksiz testleri ortadan kaldırarak daha verimli test senaryoları oluşturmaya olanak sağlar [11].

Bu test teknikleri, yazılımın her yönünü kapsamlı bir şekilde test etmeyi mümkün kılar ve yazılımın kalitesini artırmada önemli bir rol oynar. Black-box testi, kullanıcı odaklı fonksiyonları test ederken, white-box testi yazılımın iç işleyişini inceler. Grey-box testi, her iki yaklaşımın birleşimini kullanarak yazılımı daha geniş bir açıdan değerlendirir, eşleme testi ise girişlerin doğru işlendiğinden emin olmak için etkili bir yöntemdir. Her bir test tekniği, yazılımın eksikliklerini tespit etmek ve daha güvenilir, kullanıcı dostu bir yazılım geliştirmek için kullanılmalıdır.

Test senaryolarının yönetimi, yazılım test süreçlerinin etkinliğini sağlamak ve yazılımın kalitesini güvence altına almak için kritik bir öneme sahiptir. Test senaryolarının doğru bir şekilde yönetilmesi, yazılım geliştirme sürecinde hataların erkenden tespit edilmesine ve düzeltilmesine olanak tanır. Etkili bir test yönetimi, sadece testlerin düzenli bir şekilde yürütülmesiyle ilgili değildir; aynı zamanda test sonuçlarının izlenmesi, analizi ve yazılımın her yönünü kapsamlı bir şekilde değerlendirilmesi gereklidir. Testlerin düzenli ve etkili bir şekilde yürütülmesi için her aşamanın dikkatlice planlanması ve her testin amacına uygun olarak yapılması gerekir. Şekil 8' de Test planlı bileşenleri verilmiştir. Bu süreç, yazılımın her bir fonksiyonunun, özelliğinin ve kullanıcının gereksinimlerinin doğru bir şekilde test edilmesini sağlamalıdır.

Şekil 8. Test planı bileşenleri veya nitelikleri

Test senaryolarının yönetiminde ilk adım, test planlarının oluşturulmasıdır. Test planı daha önce tanımlandığı üzere, hangi testlerin yapılacağını, hangi araçların kullanılacağını, testlerin hangi sıklıkla gerçekleştirileceğini ve sonuçların nasıl izleneceğini belirlemektedir. Bu plan, test sürecinin başlangıcında yazılım geliştirme ekibi ve test ekibi arasında ortak bir anlayış oluşturur ve sürecin her aşamasında takip edilmesini sağlar. Test senaryolarının yönetimi, yazılım geliştirme süreci boyunca süreklilik gerektirdiğinden, her testin önceden belirlenmiş bir zaman çizelgesine ve sıralamaya göre yürütülmesi önemlidir. Bu yaklaşım, testlerin etkili bir şekilde yürütülmesini ve her bir testin ne zaman

gerçekleştirilmesi gerektiğini sistematik bir şekilde belirlemeyi mümkün kılar.

Test Senaryosunun Adı	Adım No.	İnceleyen		Yazar Yorumları	Yorumlar
		Yorumlar	Durum		
Rapor-21-Sonuç	1	Ön koşul eksik	Kritik	Düzeltilmedi "Ön Koşul Yok"	-----
	6	Kullanılan Daha Negatif değer	Önemsiz	Sabit	-----
	11	Pozitif Değer eksik	Kritik	Sabit	-----
	19	Daha Fazla Pozitif Değere İhtiyaç Var	Önemli	-----	-----
	20	Cümle temizlenmedi	Önemsiz	Sabit Değil	-----

Şekil 9. Test senaryosu inceleme şablonu

Testlerin izlenmesi ve yönetilmesi, her bir testin nasıl sonuçlandığını ve bu sonuçların yazılımın performansına nasıl etki ettiğini görmek açısından çok önemlidir. Yazılımın test süreçlerinde kullanılan araçlar, testlerin nasıl ilerlediğini izlemeyi ve sonuçları toplama aşamasında yönetmeyi kolaylaştırır. Bu araçlar, yazılım testlerinin sonuçlarını kaydeder, hataları tespit eder ve raporlar. Ayrıca, test sonuçlarını düzenli olarak izlemek ve bu sonuçları değerlendirmek, yazılımın hangi alanlarında iyileştirmeler yapılması gerektiğini belirler. Şekil 9' da çalışma kapsamında kullanılan Test Senaryosu İnceleme Şablonu verilmiştir. 112 acil çağrı merkezi yazılımı gibi kritik sistemlerde, test sonuçları düzenli olarak izlenmeli ve analiz edilmelidir, çünkü bu sistemlerde olası hatalar kullanıcı güvenliğini doğrudan etkileyebilir [11].

Son olarak, test sonuçlarının raporlanması ve yönetilmesi, yazılım geliştirme sürecinin önemli bir parçasıdır. Test sonuçları, yazılımın her aşamasında yapılan değişikliklerin ve düzeltmelerin ne kadar etkili olduğunu göstermek için kullanılır. Testlerin başarı oranları, hata tespit oranları ve test kapsamı gibi faktörler, yazılımın kalitesini belirlemek için değerlendirilir. Testlerin sonunda hazırlanan raporlar, yazılımın mevcut durumu hakkında ayrıntılı bilgi sunarak, yazılım geliştirme ekibine gerekli geri bildirimleri sağlar. Bu raporlar, yazılımın iyileştirilmesi gereken noktaları ve hangi alanlarda geliştirme yapılması gerektiğini ortaya koyar [3]. Şekil 10' da örnek Test Yürütme Şablonu verilmiştir.

Modül Adı	Toplam Test S.	Uygulanan Test S.	Geçen Test S.	Kalan Test S.	Kalan (%)	Geçen (%)
Toplam						

Şekil 10. Test yürütme raporu şablonu

Bu süreçlerin her biri, test senaryolarının yönetiminin ne kadar önemli olduğunu ve yazılımın kaliteli olmasının sağlanmasında nasıl etkili bir rol oynadığını gösterir. Etkili bir test senaryosu yönetimi, yazılım geliştirme sürecinin her aşamasında kaliteyi artırmaya katkı sağlar ve yazılımın güvenli, stabil ve kullanıcı dostu olmasını garanti eder.

III. GEREÇ VE YÖNTEM

Bu çalışmanın amacı, 112 acil çağrı merkezi uygulamasının test senaryolarını ve kullanıcı memnuniyetini değerlendirmektir. Bu bağlamda, kullanılan araştırma modeli nicel araştırma modelidir. Nicel araştırmalar, ölçülebilir veriler üzerinden yapılan analizlerle daha objektif sonuçlar elde etmeye olanak tanır. Bu model, yazılımın performansını, işlevselliğini ve kullanıcı memnuniyetini sayısal verilerle analiz etmek amacıyla tercih edilmiştir.

3.1. Araştırma Grubu

Bu çalışmada, Türkiye genelinde 112 acil yardım istasyonlarında görev yapan çağrı karşılayıcı, AFAD, emniyet, itfaiye, Jandarma, orman, sağlık ve sahil güvenlik personelinden oluşan toplam 3056 kişilik bir çalışma grubu ile ölçek geliştirme çalışması yürütülmüştür. Bu geniş katılımcı yelpazesi, araştırmanın bulgularının genellenebilirliğini ve sağlamlığını artırmaktadır.

Çalışmada, ölçek geliştirmenin ilk aşaması olan aday ölçek maddelerinin oluşturulması için 112 acil yardım istasyonunda görevli 40 personelden veri toplanmıştır. Açık uçlu sorular aracılığıyla elde edilen veriler, 10 maddeden oluşan bir madde havuzu oluşturulmasına imkân sağlamıştır. Ardından, bu aday ölçek maddeleri 3056 kişilik çalışma grubuna uygulanmıştır. Elde edilen verilerle açıklayıcı faktör analizi yapılmış ve maddelerin ölçeğin yapısını ne kadar iyi yansıttığı değerlendirilmiştir.

Gorsuch (1974) ve Everitt (1975) gibi araştırmacılar, güvenilir bir ölçüm için katılımcı sayısının madde sayısının en az 5 veya 10 katı olması gerektiğini önermektedir [17, 18]. Bu çalışmada, 10 maddeden oluşan bir aday ölçek formu kullanıldığından, çalışma grubunun büyüklüğü (3056 kişi) bu önerilerin her ikisini de karşılamaktadır. Bu durum, elde edilen bulguların güvenilirliğini ve istatistiksel gücünü artırmaktadır.

Veri toplama aşamasında, 112 acil çağrı merkezindeki çalışanlardan toplamda 3056 kişilik bir katılımcı grubu oluşturulmuştur. Katılımcılar arasında yaş, cinsiyet, görev pozisyonu ve yazılımı kullanma süresi gibi demografik faktörler de dikkate alınmıştır. Bu faktörler, kullanıcıların yazılımı nasıl kullandıkları ve memnuniyet düzeylerini etkileyen önemli değişkenler olarak araştırmada yer almıştır [3].

Veri toplandıktan sonra, istatistiksel analiz yöntemleri kullanılarak elde edilen veriler incelenmiştir. Kullanıcı

memnuniyeti ve yazılımın performansı arasındaki ilişkiyi analiz etmek amacıyla frekans analizi, regresyon analizi ve ANOVA gibi yöntemler kullanılmıştır. ANOVA, farklı grupların (örneğin, yaş grupları, kurumlar, cinsiyet gibi) yazılım memnuniyetine olan etkisini değerlendirmek için kullanılan güçlü bir analiz yöntemidir [5]. Bu analizler sayesinde, yazılımın performansı üzerinde etkili olan faktörler belirlenmiş ve kullanıcıların hangi özelliklerden daha fazla memnun oldukları ortaya konmuştur.

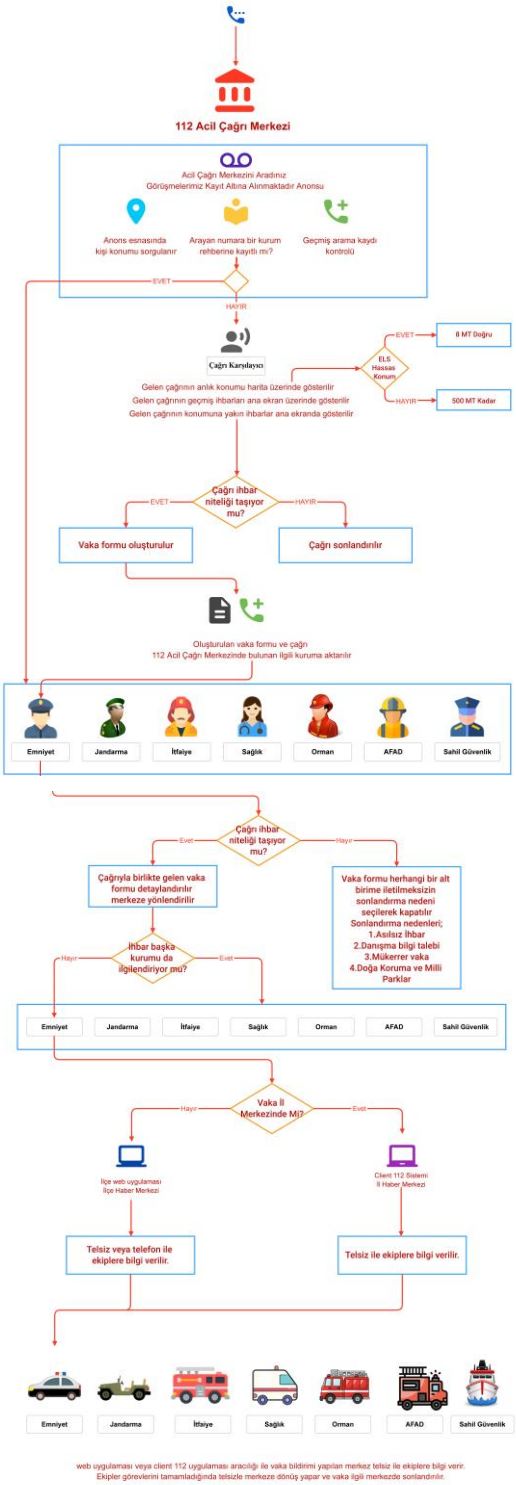
Araştırma modeli ve yöntemi, 112 acil çağrı merkezi uygulamasının işlevselliğini ve kullanıcı memnuniyetini etkileyen faktörleri kapsamlı bir şekilde analiz etmeyi hedeflemektedir. Nicel veri toplama ve analiz yöntemlerinin kullanılması, elde edilen bulguların objektif ve güvenilir olmasını sağlamaktadır.

3.2. Test Senaryolarının Oluşturulması

Test senaryoları, yazılımın doğru ve etkili bir şekilde çalıştığından emin olmak için kullanılan temel araçlardır. 112 acil çağrı merkezi uygulaması gibi kritik yazılımlarda, test senaryolarının oluşturulması, yazılımın güvenilirliğini ve işlevselliğini doğrulamak için büyük önem taşır. Test senaryoları, yazılımın her türlü kullanım senaryosuna göre tasarlanarak, yazılımın tüm potansiyel kullanım durumlarıyla uyumlu olması sağlanır.

Çalışma kapsamında oluşturulan test senaryoları, 112 Acil Çağrı Merkezi Uygulaması'nın işlevselliğini ve kullanıcı deneyimini değerlendirmek amacıyla tasarlanmış olup, bu süreçte kullanıcı geri bildirimleri ve sistemin temel fonksiyonlarının doğrulanması esas alınmıştır. Çalışma kapsamında güvenlik ve performans testleri doğrudan gerçekleştirilmemiştir; ancak anket sonuçları, sistemin kullanıcı beklentilerini ne ölçüde karşıladığını ve mevcut eksiklikleri ortaya koymada önemli bir rol oynamaktadır. Testlerin yeniden uygulanması çalışmanın mevcut kapsamı ve metodolojisi gereği mümkün olmadığından, gelecekte yapılacak araştırmalarda zorlayıcı hata senaryoları, güvenlik analizleri ve performans değerlendirmeleri gibi ek testlerin dahil edilmesi önerilmektedir. Bu sayede, sistemin yüksek trafik altındaki dayanıklılığı, hata toleransı ve güvenlik açıkları gibi kritik faktörler daha ayrıntılı olarak ele alınabilecektir.

Her test senaryosu, yazılımın belirli bir fonksiyonunu test etmeye yönelik spesifik adımlar içerir. Bu adımlar, yazılımın doğru şekilde çalışıp çalışmadığını değerlendiren bir dizi işlemi kapsar. Örneğin, 112 acil çağrı merkezi uygulamasında, kullanıcıların çağrılarını doğru birimlere yönlendirmesi, yazılımın hızlı bir şekilde yanıt vermesi ve acil durum verilerinin doğru bir şekilde işlenmesi gibi işlevler test edilen unsurlar arasında yer alır. Bu faktörler dikkate alınarak, test senaryoları oluşturulmuş ve yazılımın çeşitli koşullarda nasıl performans gösterdiği kapsamlı bir şekilde değerlendirilmiştir. Test senaryoları yalnızca yazılımın işlevselliğini değil, aynı zamanda kullanıcı deneyimini de ortaya çıkarmak için oluşturulmuştur.



Şekil 11. 112 acil çağrı uygulamasının akış şeması

Şekil 11'de 112 Acil Çağrı Merkezleri ve yeni nesil 112 uygulamasına ilişkin bir akış şeması gösterilmiştir. Akış şeması, 112 numarasına gelen bir acil durum çağrısının nasıl ele alındığını ve ilgili ekiplere nasıl yönlendirildiğini gösterir. Bu sistemde, 112 numarasına gelen acil durum çağrısı, otomatik olarak olay tipi ve konuma göre ilgili birimlere yönlendirilir. Çağrıyı değerlendiren ilgili birimler, olay yerine en uygun ekipleri sevk eder ve ekipler olay mahalline ulaşarak gerekli müdahaleyi yapar. Tüm bu süreç, çağrı merkezi tarafından izlenir ve kaydedilir.

Bu sistem sayesinde, acil durum müdahalesi önemli ölçüde hızlandırılmış ve optimize edilmiştir. Vatandaşlar tek bir numarayı arayarak yardım alabilir ve ilgili ekipler daha kısa sürede olay yerine ulaşarak gerekli müdahaleyi yapabilmektedir. Bu sayede can kaybı ve yaralanmaların önüne geçilmesi ve maddi hasarların en aza indirilmesi amaçlanmaktadır.

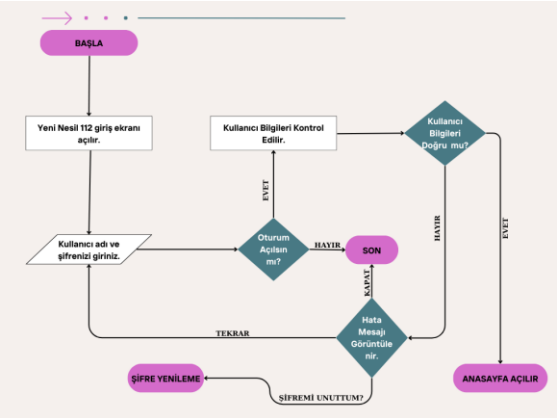
3.3. Test Senaryoları Uygulama Süreci

Bu çalışmada, 112 acil çağrı merkezi uygulamasının işlevselliğini değerlendirmek için beş farklı test senaryosu oluşturulmuş ve uygulanmıştır. İlk dört test senaryosu, kullanıcı arayüzüne yönelik testler olup uygulama üzerinde doğrudan gerçekleştirilen testlerdir. Bu testler sırasında, kullanıcıların yazılımı nasıl kullandıkları ve yazılımın her bir fonksiyonel bileşeninin nasıl çalıştığı gözlemlenmiş ve belgelenmiştir. Beşinci test senaryosu ise performans testi olup, sistemin büyük veri yükü altında nasıl çalıştığına dair veriler toplamak amacıyla yapılmıştır. Her bir test senaryosunun amacı ve kapsamı Tablo 1'de açıklanmıştır.

Tablo 1. Test senaryoları amaç ve kapsam tablosu

No	Amaç	Kapsam
T1	Kullanıcıların doğru giriş yapmalarını sağlamak	Kullanıcı adı ve şifre doğrulama işlevleri, hata mesajları ve güvenlik önlemleri.
T2	Gelen çağrıların doğru şekilde karşılanması	Çağrı numarası sorgulama, konum tespiti ve doğru birime yönlendirme
T3	Vaka formu oluşturulması ve doğru verilerin eklenmesi	Vaka bilgileri girilmesi, doğru kurumlarla paylaşılması ve takip edilmesi
T4	Vaka formunun doğru birime yönlendirilmesi	Vaka formunun doğru birime yönlendirilmesi, doğrulama yapılması
T5	Yazılım performanslarının değerlendirilmesi	Yoğun çağrı yükü altında yazılımın kesintisiz çalışması ve performans testi yapılması

Şekil 12' de Test Senaryosu 1 akış diyagramı verilmiştir. Şekil 13'de verilen Test Senaryosu 1 inceleme şablonuna uygun olarak ilk senaryo, kullanıcıların doğru kimlik bilgileriyle giriş yapmalarını sağlamak amacıyla giriş ekranının doğruluğunu test etmiştir. Kullanıcı adı ve şifre doğrulama işlevselliği kontrol edilerek, yazılımın güvenlik önlemleri ve kullanıcı doğrulama süreçleri incelenmiştir.



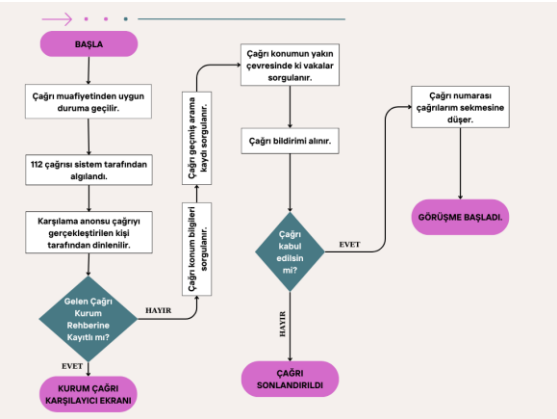
Şekil 12. Test senaryosu 1 akış diyagramı

Test Senaryosu Adı	Client-2.2.44-NG112-Giriş
Test Senaryosu Türü	Fonksiyonel Test
Gereksinim Numarası	-
Modül	Giriş
Durum	Kritik
Sürüm	2.2.44
Önkoşul	Bir giriş gerekli
Test Verisi	Kullanıcı adı – 98678220080, parola – Test01
Özet	Oturum açma işlevini kontrol etmek için

Adım No	Açıklama	Beklenen Sonuç	Gerçekleşen Sonuç	Durum	Yorumlar
1	Kullanıcı adı ve şifre girilir	Kullanıcı bilgileri girilmeli	Beklenildiği gibi	Geçti	
2	Giriş yapılır	Kullanıcı başarılı giriş yapmalı	Beklenildiği gibi	Geçti	
3	Hatalı giriş yapılırsa hata mesajı görünür	Hata mesajı görüntülenmeli	Beklenildiği gibi	Geçti	
4	Geçerli giriş sonrası ana sayfaya yönlendirilir	Ana sayfa açılmalı	Beklenildiği gibi	Geçti	

Yazar	Test Analisti
Tarih	1.1.2025

Şekil 13. Test senaryosu 1 inceleme şablonu



Şekil 14. Test senaryosu 2 akış diyagramı

Şekil 14' de akış diyagramı ve Şekil 15'de inceleme şablonu verilen şekilde karşılanıp, sistemin tüm birimlerle entegrasyonunu sağlamayı amaçlamıştır. Bu test, çağrıların doğru bir şekilde yönlendirilip, konum bilgileri gibi kritik verilerin doğru şekilde kaydedilip edilmediğini kontrol etmiştir.

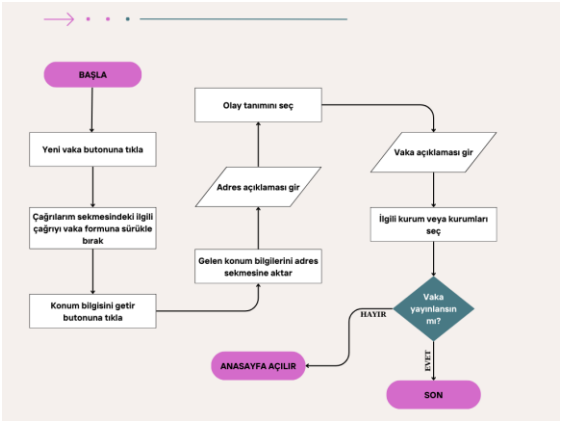
Test Senaryosu Adı	Client-2.2.44-NG112-ÇK Ana Ekranı
Test Senaryosu Türü	Fonksiyonel Test
Gereksinim Numarası	-
Modül	Çağrı karşılayıcı ana ekranı
Durum	Kritik
Sürüm	2.2.44
Önkoşul	Çağrı gerçekleştirme
Test Verisi	Telefon numarası – 0537 627 5186
Özet	Gelen çağrı bilgilerini görme

Adım No	Açıklama	Beklenen Sonuç	Gerçekleşen Sonuç	Durum	Yorumlar
1	Çağrı alınır	Çağrı başarıyla alınmalı	Beklenildiği gibi	Geçti	
2	Numara doğrulama yapılır	Numara doğrulaması yapılmalı	Beklenildiği gibi	Geçti	
3	Konum doğrulaması yapılır	Konum doğrulaması yapılmalı	Beklenildiği gibi	Geçti	
4	Çağrı yönlendirilir	Çağrı doğru birime yönlendirilir	Beklenildiği gibi	Geçti	

Yazar	Test Analisti
Tarih	2.1.2025

Şekil 15. Test senaryosu 2 inceleme şablonu

Şekil 16'da üçüncü test senaryosu verilen, çağrı karşılayıcıların yeni vaka formlarını doğru bir şekilde oluşturup, bu bilgileri doğru birimlere yönlendirmelerini test etmeyi amaçlamıştır. Şekil 17 'de şablonu verilen senaryoda, kullanıcıların vaka oluşturma sürecindeki doğruluğu, eksiksiz veri girişi yapıp yapmadıkları değerlendirilmektedir.



Şekil 16. Test senaryosu 3 akış diyagramı

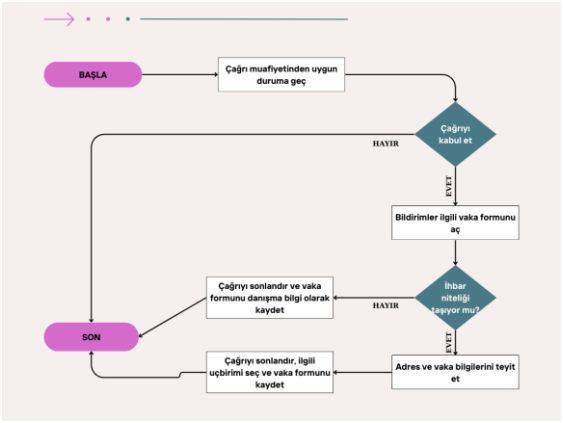
Test Senaryosu Adı	Client-2.2.44-NG112-ÇK Vaka Ekranı
Test Senaryosu Türü	Fonksiyonel Test
Gereksinim Numarası	-
Modül	Çağrı karşılayıcı vaka ekranı
Durum	Kritik
Sürüm	2.2.44
Önkoşul	Çağrı ihbar niteliği taşıyor mu?
Test Verisi	Telefon numarası – 0537 627 5186
Özet	Vaka formu oluşturma veya çağrı sonlandırma

Adım No	Açıklama	Beklenen Sonuç	Gerçekleşen Sonuç	Durum	Yorumlar
1	Yeni vaka oluşturulur	Vaka formu açılmalı	Beklenildiği gibi	Geçti	
2	Veriler girilir	Veriler doğru girilmeli	Beklenildiği gibi	Geçti	
3	Veriler doğrulanır	Veriler doğrulanmalı	Beklenildiği gibi	Geçti	
4	Vaka kaydedilir ve iletilir	Vaka ilgili kurumlara iletilmeli	Beklenildiği gibi	Geçti	

Yazar	Test Analisti
Tarih	3.1.2025

Şekil 17. Test senaryosu 3 inceleme şablonu

Şekil 18 ve Şekil 19'da verilen dördüncü test senaryosu ise çağrı yönlendiricilerin vaka formunu doğru şekilde görüntülemelerini ve vakaları ilgili birimlere iletmelerini sağlamayı hedeflemiştir. Bu test, vaka yönetimi süreçlerinin etkinliğini incelemiştir.



Şekil 18. Test senaryosu 4 akış diyagramı

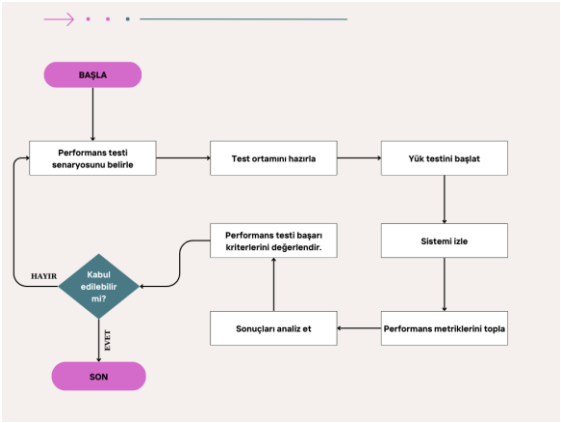
Test Senaryosu Adı	Client-2.2.44-NG112-ÇY Ana Ekranı
Test Senaryosu Türü	Fonksiyonel Test
Gereksinim Numarası	-
Modül	Çağrı yönlendirici ana ekranı
Durum	Kritik
Sürüm	2.2.44
Önkoşul	Çağrı ihbar niteliği taşıyor mu?
Test Verisi	Telefon numarası – 0537 627 5186
Özet	Çağrı ihbar niteliği taşıyor mu?

Adım No	Açıklama	Beklenen Sonuç	Gerçekleşen Sonuç	Durum	Yorumlar
1	Vaka formu alınır	Vaka formu görüntülenmeli	Beklenildiği gibi	Geçti	
2	Veriler doğrulanır	Veriler doğrulanmalı	Beklenildiği gibi	Geçti	
3	Vaka doğru birime yönlendirilir	Vaka ilgili birime yönlendirilir	Beklenildiği gibi	Geçti	

Yazar	Test Analisti
Tarih	4.1.2025

Şekil 19. Test senaryosu 4 inceleme şablonu

Beşinci ve son senaryo, uygulamanın performansını test etmek amacıyla gerçekleştirilmiştir (Şekil 20-21). Bu senaryo, yazılımın yoğun veri trafiği ve yüksek kullanıcı yükü altında nasıl performans gösterdiğini değerlendirmiş ve sistemin kesinti, donma veya yavaşlama yaşayıp yaşamadığını analiz etmiştir. Ölçülebilir kriterler çerçevesinde, testler günlük gözlenen ortalama yükün iki katı ve en kötü senaryo koşulları dikkate alınarak gerçekleştirilmiştir. Elde edilen sonuçlar, sistemin bu koşullarda güvenilirliğini ve performansını değerlendirmenin yanı sıra, kullanıcı memnuniyeti üzerindeki etkisini de ortaya koymaktadır. Eğer test bulguları, memnuniyet seviyelerinin düşmesine neden oluyorsa, bu parametrelerin iyileştirilmesi yazılım optimizasyonu açısından kritik hale gelmektedir.



Şekil 20. Test senaryosu 5 akış diyagramı

Test Senaryosu Adı	Client-2.2.44-NG112-ÇY
Test Senaryosu Türü	Performans Testi
Gereksinim Numarası	-
Modül	Çağrı yönlendirici ana ekranı
Durum	Kritik
Sürüm	2.2.44
Önkoşul	
Test Verisi	İl kod – 06, Kurum kod-1, ip-1.2.3.4, te- 11.11.11.11
Özet	Yeni vaka oluşturma, vaka yayınlama, vaka kanal yayınlama?

Adım No	Açıklama	Beklenen Sonuç	Gerçekleşen Sonuç	Durum	Yorumlar
1	Yük testi başlatılır	Yük testi sorunsuz başlatılmalı	Beklenildiği gibi	Geçti	
2	Sistem izlenir	Sistem stabil çalışmalı	Beklenildiği gibi	Geçti	
3	Performans metrikleri toplanır	Performans metrikleri doğru toplanmalı	Beklenildiği gibi	Geçti	
4	Sistem stabil mi?	Sistem stabil olmalı	Beklenildiği gibi	Geçti	

Yazar	Test Analisti
Tarih	5.1.2025

Şekil 21. Test senaryosu 5 inceleme şablonu

Bu test senaryoları, yazılımın her bir fonksiyonunun doğruluğunu ve güvenilirliğini test etmek için kapsamlı bir şekilde tasarlanmıştır. Her bir senaryo, 112 acil çağrı merkezinin işlevselliğini ve acil durum yönetimi süreçlerini değerlendirmeyi amaçlamakta olup, yazılımın gerçek dünya koşullarındaki performansını anlamak için önemlidir.

3.4. Kullanıcı Memnuniyeti Anketi

Yeni Nesil 112 Uygulaması, acil durumlarda vatandaşların hayat kurtarıcı yardım almasına yardımcı olmak için kritik bir rol oynamaktadır. Bu nedenle, uygulamanın güvenilir, kullanımı kolay ve etkili olması büyük önem taşır. Uygulamanın bu hedeflere ne ölçüde ulaştığını değerlendirmek için bir kullanıcı memnuniyeti anketi yapılmıştır. Anket, kullanıcı deneyimini ve uygulamanın işlevselliğini değerlendirmek, yazılımdaki hataları ve sorunları belirlemek, geliştirme önerilerini toplamak ve kullanıcı güvenini artırmak gibi amaçlarla kullanılmıştır.

Bu memnuniyet anketi, 112 acil çağrı uygulamasını aktif olarak kullanan çeşitli profesyonel gruplara uygulanmıştır. Katılımcılar, çağrı karşılayıcılar, çağrı yönlendiriciler ve teknik personelden oluşmaktadır. Çağrı karşılayıcılar, 112 uygulamasında ön çağrıyı karşılayan valilik personelleridir. Çağrı yönlendiriciler ise 112 acil çağrı merkezlerinde görevli emniyet, sağlık, orman, itfaiye, AFAD, jandarma, sahil güvenlik gibi kurum personelleridir. Teknik personel ise 112 acil çağrı merkezlerinde teknik destek veren valilik çalışanlarıdır.

Tablo 2' de verilen anket, kullanıcıların yazılımla olan etkileşimlerini, karşılaştıkları zorlukları ve uygulamanın etkinliğini değerlendirmeyi amaçlamaktadır. Katılımcılar, farklı illerde görev yapan personellerden rastgele seçilmiş olup, farklı cinsiyet, yaş grubu ve deneyim seviyelerine sahip

bireyleri içermektedir. Böylece, 112 acil çağrı uygulamasının çeşitli kullanıcı grupları üzerindeki etkileri geniş bir perspektiften değerlendirilmiş olmaktadır. Anket, tüm katılımcılara çevrim içi ortamda sunulmuş ve belirli bir süre içinde yanıt verilmesi istenmiştir.

Bu çalışmada, kullanıcıların verdiği toplam puan üzerinden yapılan değerlendirme, puanlar ne kadar yüksekse memnuniyetin de o kadar fazla olduğunu göstermektedir. Anket sonuçları ayrıca, katılımcıların çağrı sistemlerini kullanma süreleri, çalıştıkları iller, cinsiyetleri ve kurumlarına göre de analiz edilmiştir. Bu demografik faktörlerin memnuniyetle olan ilişkisi ise ANOVA analizi ile incelenmiştir.

Tablo 2. Kullanıcı memnuniyet anket soruları

No	Soru Metni
1	Yazılımın görsel tasarım ve kullanıcı arayüzü çalışma şartları için iyi seviyededir.
2	Yazılımın arayüzünün özelleştirilebilirliği yeterince esnek ve istenilen ekran tasarımını sağlayabilmektedir.
3	Yazılım kullanımım sırasında tüm beklentilerimi karşılamaktadır.
4	Yazılımın kullanımı sırasında herhangi bir hata ya da çökme sorunu yaşamıyorum.
5	Yazılımın kullanımı sırasında kesintiler veya bağlantı sorunları yaşamıyorum.
6	Yazılımın hızı ve performansı konusunda sorun yaşamamaktayım.
7	Yazılımın güvenliği hakkında herhangi bir endişem bulunmamaktadır.
8	Yazılımın geri bildirimlerini gönderme veya hata bildirme süreçleri iyi seviyededir
9	Yazılım ihtiyaçlarımız doğrultusunda düzenli olarak güncellenerek daha iyi hale gelmektedir.
10	Yazılımın raporlama ve istatistikleri yeterince açıklayıcı ve detaylı şekilde sunulmaktadır.

Toplam anket puanını hesaplarken 2 farklı yol izledik. Ortalama Toplam Puan (OTP) ve 100 üzerinden normalize edilmiş (OTP 100) ifadelerini kullandık.

OTP, bir grup için verilen toplam puanların ortalaması anlamına gelir. Formülü şu şekildedir:

$$OTP = \frac{\sum_{i=1}^{n} x_i}{n} \quad \textbf{(1)}$$

$\sum_{i=1}^{n} x_i$ ifadesi tüm kişilerin toplam puanlarının toplamı, n kişi sayısı, x_i i. Kişinin toplam puanı olarak anlamlandırılmıştır.

Örnek; bir ilde 5 kişi varsa ve bunların puanları 40,45,30,25,50 ise; $OTP = \frac{40+45+30+25+50}{5} = \frac{200}{5} =$ 40 olur.

OTP 100 tanımı OTP'nin 100 üzerinden normalize edilmiş halidir.

$$OTP\ 100 = \frac{\frac{\sum_{i=1}^{n} x_i}{n}}{Q \times S} \times 100 \quad \textbf{(2)}$$

Değişkinlerin anlamı: $\sum_{i=1}^{n} x_i$ tüm kişilerin toplam puanlarının toplamı, n kişi sayısı, Q toplam soru sayısı, S her sorunun maksimum puanıdır.

Örnek; Düzenlenen bir ankette 10 soru sorulmuştur ve soruların cevaplar 1-5 arasında puanlandırılmıştır. Bu ankete 7 kişi katılmış ve bunların puanları 25, 28, 30, 35, 45, 50, 10 ise $OTP\ 100 = \frac{\frac{25+28+30+35+45+50+10}{7}}{10 \times 5} \times 100 = \frac{\frac{223}{7}}{50} \times 100 = 0{,}637 \times 100 = 63{,}7 = 64$ olur.

Standart sapma, bir setin ne kadar yayıldığını ölçen bir değerdir. Yani, verilen yanıtların ortalama değerden ne kadar uzaklaştığını gösterir. Yüksek bir standart sapma, yanıtların daha fazla çeşitlilik gösterdiğini, düşük bir standart sapma ise yanıtların birbirine yakın olduğunu gösterir.

$$SS = \sqrt{\frac{\sum(X_i - Ortalama)^2}{N}} \quad \textbf{(3)}$$

Tablo 3. Kullanıcı memnuniyet anketi – OTP, OTP 100 ve Standart Sapma puanları

No	OTP	OTP 100	SS
1	3,32	66	1,01
2	3,25	65	1,02
3	2,93	59	1,06
4	2,61	52	1,12
5	2,65	53	1,11
6	3,24	65	1,13
7	3,75	75	0,97
8	3,33	67	1,05
9	3,48	70	0,99
10	3,47	70	1,01

Tablo 3'teki standart sapma değerleri, kullanıcı deneyimindeki tutarlılığı ve farklılıkları ortaya koymaktadır. Güvenlik konusunda düşük standart sapma değeri, kullanıcıların yazılımın güvenliğine dair ortak bir algıya sahip olduğunu gösterirken, performans, bağlantı stabilitesi ve hata/çökme

sorunlarında yüksek standart sapma değerleri, kullanıcı deneyimlerinde önemli farklılıklar olduğunu ortaya koymaktadır. Bu durum, yazılımın bazı kullanıcılar için sorunsuz çalışırken, bazıları için performans ve bağlantı sorunları yaşattığını göstermektedir. Bu nedenle, sistem performansının iyileştirilmesi, hata yönetimi ve yük testlerinin artırılması, kullanıcı deneyimindeki tutarsızlıkları gidermek adına kritik bir gereklilik olarak öne çıkmaktadır.

Bu çalışmada 10 sorudan oluşan bir memnuniyet anketi 3056 kişiye uygulanmış ve anketin istatiksel değerlendirmesi amacıyla IBM SPSS 26 programı kullanılmıştır. IBM SPSS 26, memnuniyet anketleri için güçlü bir araçtır ve anket verilerinizden en iyi şekilde yararlanmanıza yardımcı olur. Anketlerinizi tasarlamanıza, analiz etmenize ve raporlamanıza olanak tanıyarak, işinizi geliştirmek için değerli bilgiler elde etmenizi sağlar.

Bu çalışmada memnuniyet anketinde değerlendirme ankete verilen toplam puanın değeri ne kadar yüksek ise memnuniyet o kadar fazladır. Ayrıca çalışmaya katılan kişilerin çağrı sistemlerini kullanma yılları, çalışılan iller, cinsiyet ve kuruma göre memnuniyet anketinin toplam puanları arasındaki değişim olup olmadığı ANOVA analizi ile incelenmiştir. ANOVA analizi, istatistikte varyans analizi olarak da bilinen ve grup ortalamaları arasındaki farkların anlamlılığını test etmek için kullanılan bir yöntemdir.

ANOVA analizi, toplam karelerin farkı, serbestlik dereceleri (df), ortalama karelerin hesaplanması, F-Değeri hesaplama, P-Değeri ve sonuç adımlarından oluşmaktadır.

Ortalama hesaplama genel ortalama ve grup ortalamaları adımından oluşur.

Genel ortalama, tüm gruplardaki verilerin toplam ortalamasıdır.

$$\bar{X}_{Genel} = \frac{\sum_{i=1}^{k}\sum_{j=1}^{n_i} X_{ij}}{N} \quad \textbf{(4)}$$

k grup sayısı, n_i i-inci gruptaki eleman sayısı, X_{ij} i-inci gruptaki j-inci gözlem değeri, N toplam gözlem sayısıdır.

Grup ortalamalarında ($\bar{X}_i$) her grup için ortalanma hesaplanır.

$$\bar{X}_i = \frac{\sum_{j=1}^{n_i} X_{ij}}{n_i} \quad \textbf{(5)}$$

Toplam karelerin hesaplanması işleminde ANOVA gruplar arası (SSB), gruplar içi (SSW) ve toplam varyansı (SST) inceler.

Toplam kareler (SST) tüm verilerin genel ortalamadan sapmalarının karelerinin toplamıdır.

$$\text{SST} = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(X_{ij} - \bar{X}_{Genel})^2 \quad \textbf{(6)}$$

Gruplar arası kareler (SSB) grupların ortalamalarının genel ortalamadan sapmalarının karelerinin, her gruptaki gözlem sayısı ile çarpımının toplamıdır.

$$\text{SSB} = \sum_{i=1}^{k} n_i(X_i - \bar{X}_{Genel})^2 \quad \textbf{(7)}$$

Gruplar içi kareler (SSW) her bir grubun içindeki değerlerin, grubun kendi ortalamasından sapmalarının karelerinin toplamadır.

$$\text{SSW} = \sum_{i=1}^{k}\sum_{j=1}^{n_i}(X_{ij} - \bar{X}_i)^2 \quad \textbf{(8)}$$

Serbestlik dereceleri (df) her varyans kaynağı için hesaplanır. Bunlar;

Gruplar arası serbestlik derecesi:

$$df_{Gruplar\ Arası} = \text{k} - 1 \quad \textbf{(9)}$$

Gruplar içi serbestlik derecesi:

$$df_{Gruplar\ \text{İçi}} = \text{N} - \text{k} \quad \textbf{(10)}$$

Toplam serbestlik derecesi:

$$df_{Toplam} = \text{N} - 1 \quad \textbf{(11)}$$

Ortalama karelerin hesaplanmasında ortalama kareler, toplam karelerin serbestlik derecesine bölünmesiyle elde edilir. Bunlar;

Gruplar arası ortalama kare (MSB):

$$MSB = \frac{SSB}{df_{Gruplar\ Arası}} \quad \textbf{(12)}$$

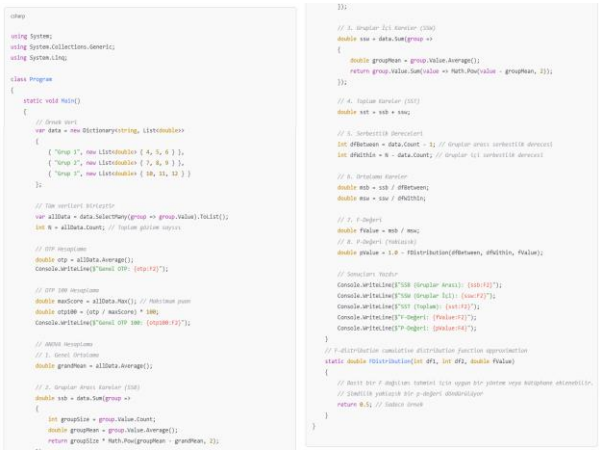
Gruplar içi ortalama kare (MSW):

$$MSW = \frac{SSW}{df_{Gruplar\ \text{İçi}}} \quad \textbf{(13)}$$

F- değeri, gruplar arası varyansın gruplar içi varyansa oranıdır.

$$F = \frac{MSB}{MSW} \quad \textbf{(14)}$$

F- değeri kullanılarak p-değeri hesaplanır. P-değeri, F-değerinin hangi olasılıkla oluştuğunu gösterir. Eğer p < 0,05 ise gruplar arasında anlamlı bir fark vardır.



Şekil 22. OTP, OTP 100 ve ANOVA hesaplamalarının C# kod örneği

IV. BULGULAR

4.1. Test Sonuçlarının Değerlendirilmesi

Yapılan test senaryolarının sonuçları, yazılımın genel işlevselliğini ve performansını değerlendirmeye yönelik önemli bulgular sağlamıştır. İlk olarak, Giriş Ekranı Testi (Şekil 13. Test Senaryosu 1) başarılı bir şekilde tamamlanmıştır. Kullanıcılar, doğru kimlik bilgileriyle giriş yapabilmiş ve hatalı girişler için doğru hata mesajları görüntülenmiştir. Ancak, hata mesajlarının daha açıklayıcı hale getirilmesi gerektiği, kullanıcıların karşılaştıkları sorunları daha rahat anlayabilmesi için bir iyileştirme alanı olarak belirlenmiştir.

Çağrı Karşılama Testi (Test Senaryosu 2), çağrıların alınması, numara doğrulama ve konum sorgulaması gibi işlemleri kapsıyordu. Bu testte de yazılım başarılı bir performans sergileyerek, tüm adımlar doğru şekilde gerçekleştirildi. Ancak, konum doğrulama işleminin bazı durumlarda zaman alıcı olabileceği, acil durumlarda hızın kritik olduğunu göz önünde bulundurarak sistemin hızının iyileştirilmesi gerektiği sonucuna varılmıştır.

Vaka Formu Oluşturma Testi (Test Senaryosu 3) de başarılı şekilde tamamlanmış, veriler doğru şekilde sisteme kaydedilmiştir. Ancak, kullanıcılar tarafından girilen verilerin doğruluğunun daha fazla denetlenmesi gerektiği, yanlış veri girişlerinin daha etkili bir şekilde engellenmesi için ek güvenlik önlemlerinin alınması gerektiği değerlendirilmiştir.

Çağrı Yönlendirme ve Vaka Formu Görüntüleme Testi (Test Senaryosu 4), vaka formunun doğru şekilde görüntülenmesi ve ilgili birime yönlendirilmesi işlemlerini kapsıyordu. Bu testte de işlem başarıyla tamamlanmış olsa da, vaka formu görüntülenirken sistemin yanıt süresinde yaşanan küçük gecikmeler, işlem hızının daha da iyileştirilmesi gerektiğini ortaya koymuştur.

Son olarak, Performans Testi (Test Senaryosu 5), yüksek kullanıcı yükü altında sistemin performansını ölçmeyi amaçlamıştı. Test başarılı bir şekilde tamamlanmış ve sistemin stabil çalıştığı gözlemlenmiştir. Ancak, daha yoğun veri trafiği altında sistem hızının düşebileceği, bu durumun ise uzun süreli kullanımda iyileştirilmesi gerektiği anlaşılmıştır.

Genel olarak, yazılımın çoğu fonksiyonu test senaryolarını başarıyla geçmiştir. Bununla birlikte, yazılımın bazı yönlerinde özellikle hız, veri doğrulama, performans ve kullanıcı geri bildirimleri konusunda iyileştirmeler yapılması gerektiği sonucuna varılmıştır. Bu öneriler doğrultusunda, yazılımın daha kullanıcı dostu, güvenilir ve verimli hale getirilmesi sağlanabilir.

4.2. Kullanıcı Memnuniyeti Anketi Analizleri

Yapılan kullanıcı memnuniyeti anketi, 112 acil çağrı uygulamasının kullanıcılar tarafından nasıl algılandığını, genel işlevselliğini ve kullanılabilirliğini değerlendirmek amacıyla önemli veriler sağlamıştır. Anketten elde edilen veriler, kullanıcı memnuniyetini etkileyen birçok faktörü belirlememize olanak tanımıştır. İlk olarak, çağrı sistemlerini kullanma süresi memnuniyet düzeyi üzerinde belirgin bir etkiye sahiptir. Uygulamanın kullanım süresi arttıkça, kullanıcıların deneyim kazandığı ve yazılımın zorluklarıyla daha iyi başa çıkabildiği gözlemlenmiştir. Ancak, Tablo 4'te verilen OTP değerleri incelendiğinde, çalışma yılı arttıkça OTP'nin azaldığı görülmektedir. Bu durum, daha uzun süreli kullanıcıların sisteme daha aşina olmasıyla birlikte hata yapma olasılıklarının düştüğünü ve süreçleri daha hızlı tamamlayabildiklerini göstermektedir. OTP'nin düşük olması, kullanıcıların daha verimli işlem yapabildiğine işaret etmektedir, dolayısıyla bu parametrenin kullanıcı memnuniyeti ile ilişkisi bağlamında değerlendirilmesi gerekmektedir. OTP değeri yıllar ilerledikçe azalmaktadır, yani daha uzun süredir uygulamayı kullanan kullanıcıların memnuniyet puanları daha düşük görünmektedir. Bu durum, kullanıcıların zamanla uygulamayla ilgili daha fazla eksiklik fark ettiğini, beklentilerinin değiştiğini veya sistemin belirli yönlerinden memnuniyetsizliklerinin arttığını gösterebilir. Özellikle yeni kullanıcıların beklentilerinin düşük olması nedeniyle başlangıçta daha yüksek puan vermesi, ancak uzun vadede sistemin eksikliklerini daha fazla deneyimledikçe puanlarını düşürdükleri görülmektedir.

Bunun yanı sıra, çalışılan il ve kurum gibi demografik faktörler de memnuniyet üzerinde etkili olmuştur. Farklı illerdeki ve kurumlardaki kullanıcılar, yazılımın sunduğu hizmetlerden farklı memnuniyet seviyelerine sahip olmuştur. Özellikle büyük şehirlerdeki kullanıcılar, yoğunluk ve altyapı sorunları nedeniyle bazı fonksiyonel zorluklar yaşarken, daha küçük

şehirlerdeki kullanıcılar genel olarak daha memnun kalmışlardır. Bu durum, yazılımın farklı bölgelerdeki ihtiyaçlara daha fazla odaklanması gerektiğini göstermektedir.

Cinsiyet faktörü de memnuniyet üzerinde önemli bir etkendir. Anket sonuçlarına göre erkek kullanıcılar, kadın kullanıcılara göre genel olarak daha yüksek memnuniyet puanları vermiştir. Bu fark, yazılımın kullanıcı deneyimi tasarımında dikkate alınması gereken bir diğer önemli faktör olarak öne çıkmaktadır. Kadın kullanıcılar, bazı özelliklerde daha fazla zorluk yaşadıklarını belirtmişlerdir, bu da yazılımın kullanıcı arayüzünde iyileştirmeler yapılması gerektiğini düşündürmektedir.

Son olarak, yazılımın genel işlevselliği ve performansı, kullanıcı memnuniyetinin en büyük belirleyicilerinden biri olmuştur. Kullanıcılar, yazılımın hızlı yanıt verme süresi, doğru veri doğrulama süreçleri ve kesintisiz çalışma gibi özelliklerini değerlendirmiştir. Bu faktörler doğrudan kullanıcı memnuniyeti ile ilişkilidir ve yazılımın daha hızlı ve hatasız çalışması gerektiği sonucunu ortaya koymuştur. Ayrıca, acil durumlarda sistemin hızının, kullanıcıların güvenliğini ve memnuniyetini artırmak için kritik bir öneme sahip olduğu anlaşılmıştır.

Genel olarak, kullanıcı memnuniyetini etkileyen faktörler arasında kullanıcı deneyimi, yazılımın hız ve doğruluğu, çalışan demografik özellikler ve yazılımın sağladığı genel işlevsellik öne çıkmıştır. Bu veriler doğrultusunda, yazılımın geliştirilmesi için önemli iyileştirme alanları belirlenmiş ve bu alanlarda yapılacak düzenlemelerle kullanıcı memnuniyetinin artırılabileceği sonucuna varılmıştır.

4.3. Çalışma Yılı, Yaş, Cinsiyet ve Kurum Gibi Faktörlerin Etkileri

Bu çalışmada çalışma yılına göre ölçek puanlarının değerlendirilmesi Tablo 4'de sunulmuştur. Tabloda çalışma yılı, ankete katılan kişi sayısı ve tüm sorulara verilen toplam anket puanı ortalama toplam puan (OTP) ve ortalama toplam puanın 100 üzerinden normalize edilmiş hali (OTP 100) olarak yer verilmiştir.

Tablo 4. Çalışma yılına göre OTP ve OTP 100

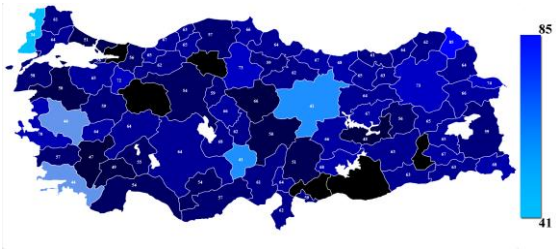
Çalışma Yılı	Kişi Sayısı	OTP	OTP 100
1 yıldan az	73	33	67
1-3 yıl arası	452	30	61
3-5 yıl arası	938	29	59
5 yıldan fazla	1.593	29	57

Tablo 4 incelendiğinde, en fazla kişi sayısı 5 yıldan fazla çalışanlar grubunda (1593 kişi) bulunurken, en az kişi sayısı ise 1 yıldan az çalışanlar grubunda (73 kişi) bulunmaktadır. Tablo 4 incelendiğinde en yüksek OTP puanı 1 yıldan az çalışanlarda görülürken en az OTP puanı ise 3 ve daha fazla çalışanlarda bulunmuştur. Çalışma yılı ile OTP arasında anlamlı fark olup olmadığı ANOVA ile incelendiğinde ise Tablo 5'deki sonuçlar elde edilmiştir.

Tablo 5. Çalışma yılına göre ANOVA analizi.

Kaynak	Karelerin Toplamı	Serbestlik Derecesi (df)	Ortalama Karesi	F	Sig. (P)
Gruplar Arası	2.501,28	3	833,76	8,43	1,41e-5
Grup içi	301.809,88	3.052	98,89		
Toplam	304.311,16	3.055			

Tablo 5 incelendiğinde çalışma yılı ile memnuniyet arasında anlamlı bir fark bulunmuştur. Çalışma yılına göre değerlendirme yapıldıktan sonra illere ve bölgelere dağılımı Şekil 24'de sunulmuştur.



Şekil 23. İllere göre OTP 100

Plaka No	Ortalama	İl nüfusu	Bölgesi	Plaka No	Ortalama	İl nüfusu	Bölgesi	Plaka No	Ortalama	İl nüfusu	Bölgesi	Plaka No	Ortalama	İl nüfusu	Bölgesi
1	31	2.283.298	Akdeniz	21	32	1.818.133	Güneydoğu Anadolu	41	Yok	2.111.907	Marmara	61	31	824.352	Karadeniz
2	34	604.978	Güneydoğu Anadolu	22	17	414.714	Marmara	42	32	2.329.241	İç Anadolu	62	34	85.366	Doğu Anadolu
3	32	751.344	Ege	23	24	604.411	Doğu Anadolu	43	29	580.701	Ege	63	yok	2.220.964	Güneydoğu Anadolu
4	33	510.626	Doğu Anadolu	24	32	245.223	Doğu Anadolu	44	33	742.725	Doğu Anadolu	64	33	375.454	Ege
5	29	338.267	Karadeniz	25	36	749.993	Doğu Anadolu	45	22	1.475.716	Ege	65	30	1.127.612	Doğu Anadolu
6	27	5.834.482	İç Anadolu	26	Yok	915.418	İç Anadolu	46	26	1.116.618	Akdeniz	66	24	418.442	İç Anadolu
7	27	2.698.249	Akdeniz	27	Yok	2.172.134	Güneydoğu Anadolu	47	32	888.874	Güneydoğu Anadolu	67	32	588.510	Karadeniz
8	31	169.403	Karadeniz	28	34	450.862	Karadeniz	48	22	1.066.736	Ege	68	31	433.055	İç Anadolu
9	29	1.161.702	Ege	29	33	145.544	Karadeniz	49	33	399.202	Doğu Anadolu	69	32	260.838	İç Anadolu
10	25	1.273.519	Marmara	30	34	275.333	Doğu Anadolu	50	31	310.011	İç Anadolu	70	37	85.012	Karadeniz
11	36	228.673	Marmara	31	31	1.544.640	Akdeniz	51	21	365.419	İç Anadolu	71	30	277.046	İç Anadolu
12	28	282.556	Doğu Anadolu	32	28	445.325	Akdeniz	52	33	775.800	Karadeniz	72	Yok	647.205	Güneydoğu Anadolu
13	32	353.988	Doğu Anadolu	33	28	1.938.389	Akdeniz	53	32	344.016	Karadeniz	73	31	557.605	Güneydoğu Anadolu
14	31	320.824	Karadeniz	34	26	15.891.924	Marmara	54	28	1.098.115	Marmara	74	32	93.481	Doğu Anadolu
15	24	273.799	Akdeniz	35	35	4.500.525	Ege	55	32	1.377.546	Karadeniz	75	42	203.351	Karadeniz
16	32	3.229.571	Marmara	36	32	274.829	Doğu Anadolu	56	34	331.311	Güneydoğu Anadolu	76	37	211.594	Doğu Anadolu
17	29	559.383	Marmara	37	29	378.115	Karadeniz	57	33	220.799	Karadeniz	77	28	296.333	Marmara
18	Yok	195.766	İç Anadolu	38	25	1.445.683	İç Anadolu	58	20	650.401	İç Anadolu	78	32	252.058	Karadeniz
19	36	524.130	Karadeniz	39	31	369.347	Marmara	59	32	1.167.059	Marmara	79	29	149.919	Güneydoğu Anadolu
20	24	1.059.682	Ege	40	33	250.519	İç Anadolu	60	31	686.934	Karadeniz	80	32	559.405	Akdeniz
												81	33	405.131	Karadeniz

Şekil 24. İllere göre OTP

Şekil 24 incelendiğinde, yeni 112 uygulamasından en çok memnun olan yer 75 plaka kodu ile Ardahan olurken en az memnuniyet ise 22 plaka ise Denizli olmuştur. Nüfusun yoğun olduğu yerlerde genellikle memnuniyet diğer nüfusu az olan illere nispetle daha az olmuştur. İller ile memnuniyet arasında ANOVA

analizi yapıldığında ise Tablo 6'deki gibi bir sonuç elde edilmiştir.

Tablo 6. Çalışılan illere göre OTP ve OTP 100.

Kaynak	Karelerin Toplamı	Serbestlik Derecesi (df)	Ortalama Karesi	F	Sig. (P)
Gruplar Arası	46.735,51	74	931,56	7,31	5,01e-65
Gruplar İçi	257.575,65	2.981	86,41		
Toplam	304.311,16	3.055			

Tablo 6 incelendiğinde F istatistik değeri: 7,31 ve bu değer gruplar arası ortalama karelerinin gruplar içi ortalama karelerine oranını göstermektedir. p değeri (Sig.): $5{,}01\,x\,10^{-65}$ değeridir. Bu, elde edilen sonuçların %5 anlamlılık düzeyinde istatistiksel olarak anlamlı olduğunu göstermektedir (çünkü $5{,}01\,x\,10^{-65}$ <0.05). Bu nedenle, null hipotezi reddedilir. Sonuç olarak, illere göre toplam ölçek puanları ortalamaları arasında istatistiksel olarak anlamlı bir farklılık olduğu görülmektedir.

Cinsiyet ile memnuniyet ortalama toplam puanlarını incelediğimizde karşımıza Tablo 7'daki gibi bir sonuç çıkmaktadır.

Tablo 7. Cinsiyete göre OTP ve OTP 100.

Cinsiyet	Kişi Sayısı	OTP	OTP 100
Kadın	1.712	28	57
Erkek	1.344	31	61

Tablo 7 incelendiğinde, erkeklerin ortalama toplam ölçek puanı 31 iken, kadınların ortalama puanı 28'dir. Bu da erkeklerin ortalama olarak kadınlardan biraz daha yüksek puan aldığını göstermektedir. ANOVA ile cinsiyet ve memnuniyet ilişkisi incelendiğinde ise Tablo 8'deki gibi bir sonuç elde edilmiştir.

Tablo 8. Cinsiyete göre ANOVA analizi.

Kaynak	Karelerin Toplamı	Serbestlik Derecesi (df)	Ortalama Karesi	F	Sig. (P)
Gruplar Arası	4.357,18	1	4.357,18	44,36	3,22e-11
Gruplar İçi	299.953,98	3.054	98,22		
Toplam	304.311,16	3.055			

Çalışma yılı, iller ve cinsiyete göre değerlendirdikten sonra kişilerin çalıştıkları kuruma göre analiz edildiğinde Tablo 9'deki gibi bir sonuç elde edilmiştir.

Tablo 9. Kurumlara göre Memnuniyet OTP ve OTP 100.

Kurum Adı	Kişi Sayısı	OTP	OTP 100
AFAD	25	32	64
Çağrı Karşılayıcı	1.377	28	55
Emniyet	393	31	62
İtfaiye	237	31	63
Jandarma	235	37	75
Orman	97	36	72
Sağlık	666	27	53
Sahil Güvenlik	26	36	72

Tablo 9'e göre jandarma kurumu en yüksek ortalama puana sahipken (37), sağlık en düşük ortalama puana sahiptir (27). Çoğu kurum 28-32 civarında puan alırken, sahil güvenlik ve orman biraz daha yüksek ortalamalara sahiptir. Kurumlarla memnuniyet anketi arasındaki ilişki incelendiğinde Tablo 10'daki sonuç elde edilmiştir.

Tablo 10. Kurumlara göre ANOVA analizi.

Kaynak	Karelerin Toplamı	Serbestlik Derecesi (df)	Ortalama Karesi	F	Sig. (P)
Gruplar Arası	31.663,36	7	4.523,34	50,57	1,85 e-68
Gruplar İçi	272.647,80	3.048	89,45		
Toplam	304.311,16	3.055			

Tablo 10 incelendiğinde F değeri 50,57 ve anlamlılık düzeyi $1{,}85\,x\,10^{-68}$ olduğundan null hipotez reddedilir. Bu da kurumlar arası ortalama toplam ölçek puanlarında istatistiksel olarak anlamlı bir farklılık olduğunu göstermektedir.

Katılımcıların yaş dağılımı incelendiğin de büyük bir kısmı 25-45 yaş aralığındadır. Bu yaş grubu, yazılımın ana kullanıcı kitlesini oluşturmaktadır. Tablo 11'de yaş aralığı ve katılımcı sayısı verilmiştir.

Tablo 11. Yaş dağılımına göre OTP ve OTP 100.

Yaş Aralığı	Kişi Sayısı	OTP	OTP 100
18-25	304	30	59
26-35	1.550	29	58
36-45	863	29	58
46-60	320	33	65
60'dan büyük	19	31	63

Tablo 11'e göre en çok personel 26-35 yaş aralığında bulunurken 26-45 yaş aralığında 29 puanla en düşük puana sahiptir. En yüksek puan 46-60 yaş aralığındayken en az personel 19 kişiyle 60'dan büyük aralığındadır.

Tablo 12. Yaş dağılımına göre ANOVA analizi.

Kaynak	Karelerin Toplamı	Serbestlik Derecesi (df)	Ortalama Karesi	F	Sig.
Gruplar Arası	22,94	2.991	4,59	8,500	0
Gruplar İçi	1.645,81	2.991	0,54		
Toplam	1.668,75	3.056			

Tablo 12 incelendiğinde gruplar arası, yaş grupları arasındaki varyans, karelerin toplamı 22,94 ve F istatistiği 8,5 hesaplanmıştır. Gruplar içi, her taş grubu içindeki varyans karelerin toplamı 1645,81 ve ortalama karesi 0,54 hesaplanmıştır. F istatistiği 8,5 olarak hesaplanmıştır ve bu değer P değeri 0,0 ile anlamlı bir fark olduğunu göstermektedir. Bu sonuçlarla yaş aralığına göre gruplar arasında farkın anlamlı olduğunu görebiliriz.

4.4. Test Süreçlerinin Kısıtlılıkları ve İyileştirme Önerileri

Yapılan test süreçleri, 112 acil çağrı uygulamasının çeşitli işlevlerini ve kullanıcı etkileşimini değerlendirmek açısından önemli bilgiler sağlamış olsa da bazı kısıtlılıklar mevcuttur. Bu kısıtlılıklar, testlerin gerçek dünya senaryolarını yeterince yansıtamaması ve bazı kritik test aşamalarının eksik kalması gibi unsurları içermektedir. Bu nedenle, test süreçlerinin daha etkin hale getirilmesi için belirli iyileştirmeler önerilmektedir.

Birincil kısıtlama, test senaryolarının gerçek acil durum koşullarını tam anlamıyla yansıtamamasıdır. Çoğu test, sistemin temel işlevselliğini kontrol etmeye odaklanmışken, acil durumların karmaşık doğasını ve kullanıcıların yüksek stres altındaki performansını simüle edememiştir. Bu nedenle, test senaryolarının daha gerçekçi hale getirilmesi, yazılımın acil durum senaryolarındaki etkinliğini daha doğru şekilde ölçebilmek için önemlidir. Gelecekteki testlerde, daha karmaşık senaryolar ve çoklu acil durum senaryoları kullanılarak yazılımın dayanıklılığı daha iyi test edilebilir. Ayrıca, yazılımın stresli durumlarla nasıl başa çıktığını test etmek için yük testlerinin ve stres testlerinin daha fazla vurgulanması gerekir.

Bir diğer kısıtlama, kullanıcı çeşitliliği ve farklı demografik özelliklere sahip kullanıcılar üzerindeki etkilerin yeterince dikkate alınmamış olmasıdır. Testlerde genellikle belli bir grup kullanıcıya odaklanılmışken, farklı yaş grupları, cinsiyetler ve deneyim seviyelerindeki kullanıcıların yazılımı nasıl kullandığı yeterince araştırılmamıştır. Test süreçlerine daha geniş bir kullanıcı kitlesi dahil edilerek, yazılımın farklı demografik özelliklere sahip bireyler üzerindeki etkisi daha kapsamlı bir şekilde analiz edilmelidir. Bu tür kullanıcı profillerinin teste dahil edilmesi, yazılımın genel erişilebilirliğini artıracaktır.

Ayrıca, otomatik testlerin ve manüel testlerin dengeli bir şekilde entegrasyonu eksik kalmıştır. Manüel testler, testlerin doğruluğunu kontrol etme açısından faydalı olsa da otomatik testler, yazılımın belirli işlevlerini daha hızlı ve tekrar edilebilir şekilde test etme imkânı sağlar. Öneri: Otomatik test araçlarının kullanımının artırılması, yazılımın daha hızlı ve sürekli olarak test edilmesini sağlayacaktır. Otomatik testler, özellikle yazılımın performansını ve güvenliğini kontrol etmek için etkili olabilir.

Eğitim ve test senaryoları doğrulama da önemli bir eksiklik olarak öne çıkmaktadır. Test sürecine katılan bazı kullanıcılar, yazılımın tüm işlevleri ve test senaryoları hakkında yeterince bilgi sahibi olmamış olabilir. Bu durum, testlerin doğruluğunu ve geçerliliğini etkileyebilir. Öneri: Test katılımcılarının eğitimlerinin daha kapsamlı olması, ayrıca test senaryolarının daha detaylı şekilde doğrulanması, yazılımın işlevsellik testlerinin daha güvenilir olmasını sağlar.

Son olarak, test sonrası geri bildirim ve analiz süreçleri yeterince derinlemesine yapılmamıştır. Testlerden elde edilen verilerin analiz edilmesi ve yazılımın zayıf yönlerinin belirlenmesi aşamasında eksiklikler bulunmaktadır. Öneri: Test sonrası geri bildirimlerin toplanması ve düzenli bir analiz sürecinin oluşturulması, yazılım geliştirme sürecinde iyileştirme alanlarının daha hızlı bir şekilde belirlenmesini sağlayacaktır.

Bu iyileştirmeler, test süreçlerinin daha kapsamlı, etkin ve gerçekçi olmasını sağlayarak yazılımın kalitesini artıracaktır. Ayrıca, yazılımın kullanıcı dostu ve güvenilir olma hedeflerine daha etkili bir şekilde ulaşılmasını sağlayacaktır.

V. TARTIŞMA VE SONUÇ

Bu çalışmada elde edilen bulgular, 112 acil çağrı uygulamasının kullanıcı memnuniyeti ve yazılım testleri üzerine yapılan testlerin etkinliğini ortaya koymaktadır. Yazılımın performansı, kullanıcı etkileşimi ve işlevselliği değerlendirilmiş olup, bu değerlendirmeler mevcut literatürdeki benzer çalışmalarla karşılaştırıldığında bazı önemli benzerlikler ve farklılıklar gözlemlenmiştir.

Öncelikle, kullanıcı memnuniyeti ile ilgili elde edilen sonuçlar, önceki çalışmalarda belirtilen bulgularla uyum göstermektedir. Örneğin, acil çağrı uygulamalarının kullanıcı memnuniyetini artırmak için yazılımın basit, hızlı ve güvenilir olması gerektiğini vurgulamaktadır [19, 20]. Benzer şekilde, bu çalışmada da kullanıcıların uygulama üzerinden aldıkları hizmetin hızını ve güvenilirliğini olumlu bir şekilde değerlendirdikleri görülmüştür. Kullanıcıların hızlı ve doğru bilgi edinme, çağrıları etkili bir şekilde yönlendirme ve acil durumlarla etkin şekilde başa çıkabilme kabiliyetine sahip olmaları, kullanıcı memnuniyetini artıran temel unsurlar olarak öne çıkmıştır.

Ancak, mevcut literatürle karşılaştırıldığında, bu çalışmada bazı test süreçlerinin daha derinlemesine ele alınması gerektiği sonucuna varılmıştır. Örneğin, acil çağrı yazılımlarının performans testlerinin daha kapsamlı bir şekilde yapılması gerektiğini belirtirken, bu çalışmada, stres testlerinin daha fazla yer alması gerektiği öne çıkmıştır [21]. Bu tür testler, yazılımın kriz anlarında nasıl tepki verdiğini gözler önüne serebilir ve sistemin ne kadar güvenilir olduğunu daha doğru bir şekilde değerlendirebilir. Literatürde sıkça karşılaşılan başka bir bulgu, kullanıcı etkileşimlerinin genellikle yaş ve deneyime göre değişkenlik gösterdiği ve yazılımın her kullanıcı kitlesi için özelleştirilmesi gerektiğidir [23]. Bu çalışmada da benzer şekilde, kullanıcıların yaş, deneyim ve cinsiyet gibi demografik özelliklerine bağlı olarak yazılımdan aldıkları verim ve memnuniyetin farklılık gösterdiği tespit edilmiştir.

Bunun yanı sıra, yazılım test senaryolarının doğruluğunu artırmak için sürekli güncellenmesi gerektiğini vurgulamaktadır [22]. Bu çalışma da benzer şekilde, test senaryolarının, özellikle farklı acil durum senaryolarına ve güncel teknolojik gelişmelere göre dinamik olarak güncellenmesinin gerekliliğini ortaya koymuştur. Bu, yazılımın güvenliğini ve etkinliğini artıracak ve gerçek acil durumlarda daha etkin bir şekilde çalışmasını sağlayacaktır.

Sonuç olarak, elde edilen bulgular, mevcut literatürdeki genel eğilimlerle uyumlu olmakla birlikte, test süreçleri ve kullanıcı memnuniyeti analizlerinde yapılacak bazı iyileştirmeler ile yazılımın daha da etkin hale getirilebileceğini göstermektedir. Bu çalışma, özellikle kullanıcı etkileşimlerinin daha kapsamlı analiz edilmesi gerektiğine dair literatüre katkı sağlamakta olup, gelecekte yapılacak çalışmalar için önemli bir referans oluşturacağı düşünülmektedir.

Araştırma, 112 Acil Çağrı Uygulamasının etkinliğini ve kullanıcı memnuniyetini değerlendirmeyi amaçlamıştır. Yapılan test senaryoları ve memnuniyet anketleri, uygulamanın güçlü ve zayıf yönlerini ortaya koymuş ve yazılımın geliştirilmesi için önemli ipuçları sağlamıştır. Test süreçleri, yazılımın fonksiyonel işlevselliğini doğrulamakla kalmayıp, aynı zamanda kullanıcı deneyimini ve performansını da kapsamlı bir şekilde değerlendirmiştir. Elde edilen bulgulara göre, yazılımın temel işlevsellik açısından iyi performans sergilediği, ancak bazı alanlarda, özellikle kullanıcı arayüzü ve sistem entegrasyonları gibi konularda iyileştirmelere ihtiyaç duyulduğu belirlenmiştir.

Yapılan analizlerde, test senaryolarının çoğunlukla başarılı olduğu, ancak daha karmaşık acil durum senaryoları ve çoklu kullanıcı etkileşimleri gibi daha geniş kapsamlı testlerin eklenmesi gerektiği sonucuna varılmıştır. Ayrıca, kullanıcı memnuniyeti anketleri, yazılımın kullanıcı dostu ve etkin olduğuna dair genel bir görüş ortaya koymuş olsa da bazı kullanıcı gruplarının, özellikle acil çağrı merkezi çalışanlarının, yazılımın kullanımında belirli zorluklarla karşılaştığı tespit edilmiştir.

Genel olarak, bu araştırma, acil çağrı uygulamaları konusunda yapılacak daha fazla çalışmanın gerekliliğini ve bu tür sistemlerin kullanıcı memnuniyeti ile güvenliğini artırma potansiyelini ortaya koymuştur. Sonuçlar, 112 Acil Çağrı Uygulamasının, güncel test süreçleri ve kullanıcı deneyimi analizi ışığında daha verimli ve etkin bir şekilde geliştirilmesine yönelik yönlendirmeler sunmaktadır.

Gelecekte yapılabilecek çalışmalar, 112 Acil Çağrı Uygulaması ve benzeri acil durum yazılımlarının daha etkin ve güvenli hale getirilmesi için kritik öneme sahiptir. İlk olarak, yazılımın performansını ve dayanıklılığını test etmek amacıyla daha geniş ve farklı senaryoları kapsayan test süreçlerinin uygulanması önerilmektedir. Özellikle, yazılımın yüksek kullanıcı yoğunluğu altında nasıl performans gösterdiği, acil durumlar sırasında çoklu kullanıcı etkileşimlerinin yazılım üzerindeki etkileri gibi alanlar daha derinlemesine incelenmelidir. Bu tür testler, yazılımın olası aksaklıklarını önceden tespit etmek ve müdahale süreçlerini iyileştirmek adına önemlidir.

Bir diğer önemli nokta, yazılımın kullanıcı deneyimini geliştirmeye yönelik yapılacak çalışmaların artırılmasıdır. Kullanıcı arayüzü ve etkileşimli süreçlerin daha sezgisel ve anlaşılır hale getirilmesi, acil durumlarda daha hızlı ve doğru müdahaleyi mümkün kılacaktır. Gelecekteki çalışmalarda, özellikle acil çağrı merkezi çalışanlarının yazılımla olan etkileşimleri üzerinden yapılacak gözlemler, yazılımın kullanıcı dostu olup olmadığını değerlendirmek için kritik veri sağlayacaktır.

Ayrıca, sistem entegrasyonları ve farklı kurumlarla olan uyum konusunda yapılacak araştırmalar da önemlidir. 112 Acil Çağrı Uygulaması, sağlık, güvenlik ve itfaiye gibi farklı kurumlarla entegre çalıştığı için bu entegrasyon süreçlerinin kesintisiz ve doğru bir şekilde sağlanması gerekmektedir. Gelecekte yapılacak çalışmalar, yazılımın bu kurumlarla daha etkili iletişim kurmasını sağlayacak altyapıların geliştirilmesi yönünde odaklanmalıdır.

Son olarak, kullanıcı memnuniyeti ve geri bildirimlerin yazılım geliştirme süreçlerine daha fazla dahil edilmesi gerektiği bir diğer önemli bulgudur. Gelecekte yapılacak anketler, gözlemler ve kullanıcı geribildirimlerini toplayarak, yazılımın performansı ve kullanıcı deneyimi hakkında daha fazla veri toplanabilir. Bu sayede, yazılımın kullanıcı ihtiyaçlarına göre daha fazla özelleştirilmesi ve sürekli olarak iyileştirilmesi sağlanabilir.

Bu öneriler ışığında, 112 Acil Çağrı Uygulamasının daha güvenli, verimli ve kullanıcı dostu hale getirilmesi adına çalışmaların genişletilmesi, yenilikçi teknolojilerin entegrasyonu ve kapsamlı testlerin yapılması gerektiği açıkça görülmektedir. Gelecekte yapılacak araştırmalar, acil durum çağrı sistemlerinin etkinliğini ve güvenliğini artırmak adına önemli adımlar atılmasına olanak sağlayacaktır.

YAZAR KATKI BEYANI

Gerçekleştirilen çalışmada Yusuf SELİMDAROĞLU fikrin oluşması, tasarımın yapılması, literatür taraması, çalışma kapsamında geliştirilen ölçeğin uygulanması, elde edilen sonuçların değerlendirilmesi başlıklarında; Özgür TONKAL sonuçların incelenmesi, içerik açısından makalenin kontrol edilmesi başlıklarında katkı sunmuşlardır.

ETİK KURUL ONAYI VE ÇIKAR ÇATIŞMASI BEYANI

Bu çalışma kapsamında, Samsun Üniversitesi Etik Kurulu (Etik kurul onay numarası: 2024-76) tarafından onay alınmıştır. Ayrıca, anketin gerçekleştirildiği İller İdaresi Genel Müdürlüğünden yazılı izin alınmış ve anket çalışması, kurum personellerine etik kurallar çerçevesinde uygulanmıştır.

KAYNAKLAR

[1] Aydemir, E., Demirel, N.Ç., Kıvrak, T., & Yüksel, M. (2014). Development of an Artificial Neural Networks Prediction Model for Determining Call Numbers to 112 Emergency Call Center. Pamukkale University Journal of Engineering Sciences, 20(5), 145-149.

[2] Alpakan, A.H. (2019). Evaluation of Factors Affecting Attention Level of 112 Emergency Call Center Employees. Institute of Health Sciences.

[3] Kaya, Z., & Öcal, H. (2023). The Effect of Transformational Leadership on Organizational Trust in Virtual Organizations - A Research in 112 Emergency Call Centers. Prehospital Journal, 8(1), 13-29.

[4] Tarcan, M., Hikmet, N., Schooley, B., Top, M., & Tarcan, G.Y. (2017). An Analysis of the Relationship Between Burnout, Socio-demographic and Workplace Factors and Job Satisfaction Among Emergency Department Health Professionals. Applied Nursing Research, 34, 40-47.

[5] Yuldoşev, D. (2015). The Effect of Technology Literacy on Electronic Product Purchasing Behavior (Kyrgyzstan Example). Sakarya University.

[6] Uysal, İ., Kıyak, M., Eryılmaz, M., & Bektaş, G. (2022). Development and Application Example of Evaluation Scale in Terms of Satisfaction, Efficiency, Safety and Performance of 112 Emergency Aid Stations. Prehospital Journal, 7(2), 159-174.

[7] Subudhi, B.S.K., Trecarichi, G., Dragoni, M., & Baillie, J.K. (2020). Performance Testing for VoIP Emergency Services: A Case Study of the EMYNOS Platform and a Reflection on Potential Blockchain Utilisation for NG112 Emergency Communication. Journal of Ubiquitous Systems & Pervasive Networks, 12(1), 1-8.

[8] Siddiky, M.N.A., Rahman, M.E., & Uzzal, M.S. (2024). Beyond 5G: A Comprehensive Exploration of 6G Wireless Communication Technologies.

[9] Yoou, S.K., & Kwon, H.J. (2015). Study About the Satisfaction with Simulation Practice Course Experience on ACLS of Paramedic Students. Journal of The Korea Academia-Industrial Cooperation Society, 16(10), 6647-6654.

[10] Serdaroğlu, D. (2015). Software Testing Process and PRISMA Approach Application in TMMi Model. Institute of Science and Technology.

[11] Garousi, V., Shepherd, D.C., & Herkiloğlu, K. (2020). Software-testing Education: A Systematic Literature Mapping. Journal of Systems and Software, 165, 110570.

[12] Özdemir, E. (2020). Agile Approaches in Project Management in Large-scale Companies. Işık University.

[13] Alshamrani, A., & Bahattab, A. (2015). A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model. International Journal of Computer Science Issues, 12(1), 106.

[14] Nalbant, E. (2020). The Importance of Testing in Software Life Cycle and Implementation of a Test Automation. Institute of Science and Technology.

[15] Başar, A. (2015). Improvement of Test Processes with Maturity Level Model: An Application in a Software Development Company with Scrum. XVII. Academic Informatics Conference (AB 2015).

[16] Öztürk, İ. (2022). The Importance and Methods of Software Testing Processes in Banking Sector. Journal of Computer Sciences and Technologies, 3(1), 23-29.

[17] Yılmaz, N. (2023). Sustainable Leadership: A Scale Adaptation Study. Marmara University.

[18] Bektaş, H. (2015). Factor Analysis for Binary Variables: An Application on Quality of Work Life.

[19] Al-Hajri, F., & Al-Shaibani, A. (2019). Evaluating Emergency Management Systems: Key Performance Indicators and User Satisfaction. International Journal of Emergency Management, 18(3), 151-165.

[20] Jørgensen, H. (2020). User Satisfaction in Emergency Response Systems: A Case Study of 112 Emergency Calls. Journal of Public Safety Technology, 34(2), 115-130.

[21] Kouadio, A., N'Guessan, L., & Kone, Y. (2021). Performance Evaluation of Emergency Call Systems Under Stress Testing Conditions. Safety Science, 138, 105188.

[22] Sharma, S., & Gupta, R. (2021). The Importance of Continuous Software Testing for Emergency Call Systems: A Review of Existing Practices. Journal of Software Engineering and Applications, 14(4), 244-259.

[23] Todorov, E., Menzel, J., & Smith, T. (2020). User Interface Design for Emergency Response Systems: The Role of Demographics in User Satisfaction. International Journal of Human-Computer Interaction, 36(7), 626-637.