T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

OTOMASYON İLE MOBİL YAZILIMLARIN TESTLERİNİN FİZİKSEL VE SANAL CİHAZLAR ÜZERİNDE GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

SÜLEYMAN ARİF ERDEM

Tez Danışmanı
Dr. Öğr. Üyesi ENGİN ŞAHİN

ÇANAKKALE – 2022

T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

BİLGİSAYAR MÜHENDİSLİĞİ ANABİLİM DALI

OTOMASYON İLE MOBİL YAZILIMLARIN TESTLERİNİN FİZİKSEL VE SANAL CİHAZLAR ÜZERİNDE GERÇEKLEŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Süleyman Arif ERDEM

Tez Danışmanı
Dr. Öğr. Üyesi ENGİN ŞAHİN

ÇANAKKALE – 2022

T.C.
ÇANAKKALE ONSEKİZ MART ÜNİVERSİTESİ
LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

Süleyman Arif ERDEM tarafından Dr. Öğr. Üyesi Engin ŞAHİN yönetiminde hazırlanan ve 26/08/2022 tarihinde aşağıdaki jüri karşısında sunulan "**Otomasyon ile Mobil Yazılımların Testlerinin Fiziksel ve Sanal Cihazlar Üzerinde Gerçekleştirilmesi**" başlıklı çalışma, Çanakkale Onsekiz Mart Üniversitesi Lisansüstü Eğitim Enstitüsü **Bilgisayar Mühendisliği Anabilim Dalı**'nda **YÜKSEK LİSANS TEZİ** olarak oy birliği ile kabul edilmiştir.

Jüri Üyeleri	**İmza**
Dr. Öğr. Üyesi Engin ŞAHİN (Danışman)	……
Prof. Dr. Mehmet Ali SALAHLI	……
Dr. Öğr. Üyesi Samet MEMİŞ	……

Tez No : 10494691
Tez Savunma Tarihi : 26/08/2022

……
Doç. Dr. Yener PAZARCIK
Enstitü Müdürü

…./…./2022

ETİK BEYAN

Çanakkale Onsekiz Mart Üniversitesi Lisansüstü Eğitim Enstitüsü Tez Yazım Kuralları'na uygun olarak hazırladığım bu tez çalışmasında; tez içinde sunduğum verileri, bilgileri ve dokümanları akademik ve etik kurallar çerçevesinde elde ettiğimi, tüm bilgi, belge, değerlendirme ve sonuçları bilimsel etik ve ahlak kurallarına uygun olarak sunduğumu, tez çalışmasında yararlandığım eserlerin tümüne uygun atıfta bulunarak kaynak gösterdiğimi, kullanılan verilerde herhangi bir değişiklik yapmadığımı, bu tezde sunduğum çalışmanın özgün olduğunu, bildirir, aksi bir durumda aleyhime doğabilecek tüm hak kayıplarını kabullendiğimi taahhüt ve beyan ederim.

Süleyman Arif ERDEM
26/08/2022

TEŞEKKÜR

Bu tezin gerçekleştirilmesinde ana etken olan, çalışmam boyunca benden bir an olsun yardımlarını esirgemeyen saygıdeğer danışman hocam Dr. Öğr. Üyesi Engin ŞAHİN'e, hayatımın her evresinde bana destek olan saygıdeğer babam Hakkı ERDEM'e, saygıdeğer annem Diçle ERDEM'e ve çok sevdiğim kardeşim Av. Neslihan ERDEM'e sonsuz teşekkürlerimi sunarım.

Süleyman Arif ERDEM

Çanakkale, Ağustos 2022

ÖZET

OTOMASYON İLE MOBİL YAZILIMLARIN TESTLERİNİN FİZİKSEL VE SANAL CİHAZLAR ÜZERİNDE GERÇEKLEŞTİRİLMESİ

Süleyman Arif ERDEM
Çanakkale Onsekiz Mart Üniversitesi
Lisansüstü Eğitim Enstitüsü
Bilgisayar Mühendisliği Anabilim Dalı Yüksek Lisans Tezi
Danışman: Dr. Öğr. Üyesi Engin ŞAHİN
26/08/2022, 52

Mobil cihazlar ve uygulamalar gelişmekte olan teknoloji ile hayatımızın her alanına girmekte, bununla beraber günlük yaşantımızın bir parçası haline gelmektedirler. Cep telefonları, tabletler, akıllı saatler, akıllı ev sistemleri gibi teknolojiler geliştikçe, geliştirilen mobil uygulamaların sayısı da her geçen gün artmaktadır. Bilişim teknolojilerinin her alanındaki uygulamalarda olduğu gibi mobil uygulamaların yazılım testleri de uygulamalar ve kullanıcılar için hayati önem arz etmektedir. Mobil uygulamalar otomatik veya manuel olarak test edilebilmektedirler. Eğer uygulama otomatik olarak test edilecekse kullanılacak olan cihazın gerçek cihaz mı yoksa sanal cihaz mı olacağı sorusuna doğru cevap vermek çok önemlidir. Otomasyon testi yeni ve gelişmekte olan bir yazılım test yöntemi olduğundan dolayı ulusal ve uluslararası düzeyde otomasyon yazılım testiyle ilgili yöntemler ve kaynaklar oldukça kısıtlıdır. Literatürdeki çalışmalar birbirinden farklı markalarca üretilen otomasyon test araçlarının karşılaştırılması şeklindedirler. Günümüzde bu alanda araştırmacıların ilgisi gittikçe artmaktadır. Sanal cihazlarda ve gerçek cihazlarda otomasyon testi yapmanın yazılım testinde sonuca etkisini ölçmek, yazılım test aracının doğru seçimi kadar önem arz etmektedir. Bu çalışmada, birden çok cihazın hem kendisiyle hem de sanal bir simülasyonu ile test çalışmaları otomasyon ile yapılarak, test sonuçları ve testler arasındaki farklar ortaya konulmaktadır. Yazılım testinde sanal ve gerçek cihazların kullanımlarının avantajları ve dezavantajları tanımlanmaktadırlar. Bu çalışma, yazılım ve bilgisayar mühendislerine otomasyon test projelerinde kullanmaları gereken cihazlara karar vermeleri konusunda yardımcı olacaktır.

Anahtar Kelimeler: Test Otomasyonu, Yazılım Testi, Sanal Cihazlar, Gerçek Cihazlar, Appium

ABSTRACT

REALIZATION OF MOBILE SOFTWARE TESTS ON PHYSICAL AND VIRTUAL DEVICES WITH AUTOMATION

Süleyman Arif ERDEM

Çanakkale Onsekiz Mart University

School of Graduate Studies

Master of Science Thesis in Computer Engineering

Supervisor: Dr. Öğr. Üyesi Engin ŞAHİN

08/26/2022, 52

Mobile devices and applications enter every aspect of our lives with the developing technology, and also become a part of our daily life. As technologies such as mobile phones, tablets, smart watches and smart home systems develop, the number of mobile applications is increasing day by day. As with applications in every field of information technology, software testing of mobile applications is of vital importance for applications and users. Mobile applications can be tested automatically or manually. If the application is to be tested automatically, it is important to correctly answer the question of whether the device to be used will be a real device or a virtual device. Since automation testing is a developing software testing method, the methods and the resources related to automation software testing at national and international level are very limited. The existing studies are generally in the form of comparison of automation test tools produced by different brands. Nowadays, the interest of researchers in this field is increasing. Measuring the effect of automation testing on virtual and real devices on the results in software testing is as important as choosing the right software testing tool. In this study, the differences between test results and tests are revealed by performing test studies with automation, both with itself and with a virtual simulation of multiple devices. The advantages and disadvantages of using virtual and real devices in software testing are defined. This study will assist software and computer engineers in deciding which devices they should use in their automation test projects.

Keywords: Test Automation, Software Testing, Virtual Devices, Real Devices, Appium

İÇİNDEKİLER

TABLOLAR DİZİNİ

ŞEKİLLER DİZİNİ

BİRİNCİ BÖLÜM
GİRİŞ

Test; Türk Dil Kurumu sözlüğüne göre "Bir kimsenin, bir topluluğun doğal veya sonradan kazanılmış yeteneklerini, bilgi ve becerilerini ölçmeye ve anlamaya yarayan sınama" olarak nitelendirilmekte, genel olarak üretilmiş herhangi bir ürünün kalitesinin kontrol edilmesi ve varsa hatalarının gün yüzüne çıkarılması için yürütülen süreçlerin genel ismidir ("Türk Dil Kurumu Sözlükleri," 2022). Testler yapılırken, üründe vaat edilen her şeye detaylıca bakılarak beklentileri karşılama derecesi ölçülür. Yazılımlar da bilgisayar ortamında üretilen ürünlerdir. Her ürün gibi yazılımların da kalitesi, vaatleri ve birer sonuç çıktıları vardır. Yazılım yazılırken yapılmak istenen ve yazıldıktan sonra alınan sonuç ne derece benzer, vaat ettiği özellikleri karşılayabiliyor mu gibi sorulara cevap alabilmek için yazılımların da test edilmesi gerekir.

Mobil cihazlar ve uygulamalar gelişen teknoloji ile hayatımızın her alanına girmiş, günlük yaşantımızın bir parçası haline gelmişlerdir. Cep telefonları, tabletler, akıllı saatler, akıllı ev sistemleri gibi teknolojiler geliştikçe geliştirilen mobil uygulamaların sayısı da her geçen gün artmaktadır. Sensor Tower şirketinin yayınlamış olduğu rapora göre (2022) 2022'nin ilk çeyreğinde dünya genelinde toplam 36,9 milyar mobil uygulama indirilmiştir. Bu uygulamaların başında içerik paylaşma platformları, çevrimiçi mesajlaşma uygulamaları alışveriş uygulamaları ve dizi, film ve müzik uygulamaları yer almaktadır (TikTok 180 milyon, Instagram 160 milyon, Facebook 155 milyon, WhatsApp 155 milyon vb.) ("Sensor Tower," 2022).

Bu yazılımlarda bulunan hatalar bağlantıların kapatılamaması, yanlış döngü kullanımları gibi bazen uygulamaların çökmesi gibi uygulama üreticilerinin istemedikleri sonuçlara sebep olurken bazen de SQL enjeksiyon için gerekli güvenlik kodların olmaması ya da eksik olması gibi sebebiyle kullanıcıların verilerin çalınmasına varan önemli problemlere neden olmaktadırlar. Bu yüzden yazılımların olabildiğince hatasız hale getirilmesi için yazılım testleri oldukça önemlidir.

Mobil uygulamaların kalitesini artırmak için mobil uygulama testlerine yönelik ilgi ve araştırmalar artmıştır (Zhan ve Yan, 2019). Artan mobil cihaz çeşitliliği, eskiden elle

(manuel) yapılan testlerin yerine artık otomasyon testlerine geçme ihtiyacı doğurmuştur (Gao vd., 2014). Otomasyonla yazılımlar test edilirken artan mobil cihaz çeşitliliği firmaların test için cihaz satın alım maliyetlerini arttırmaktadır. Artan maliyetler göz önüne alındığında bilgisayarda sanal bir telefon oluşturup (emülatör) uygulamaları bu sanal cihazlarda test etmek de alternatif bir seçenektir (Kaymak, 2020).

Bu tez çalışmasında öncelikle yazılım geliştirme yaşam döngüsü, yazılım test yaşam döngüsü, yazılım test teknikleri ve yöntemleri hakkında detaylı literatür çalışması yapılmıştır. Kullanılan uygulamalar hakkında bilgiler verilmiştir. Firmaların maliyet düşürmek için kullanabileceği bir alternatif olan sanal cihazlarda test tekniği ile gerçek cihazlarda test tekniği açıklanmıştır. Herkesin erişimine açık bir mobil uygulama olan Traffic Rider oyunu üzerinde IntelliJ IDEA uygulaması kullanılarak otomasyon testi yazılmıştır. Üç gerçek cihaz ve bu cihazların bilgisayarda oluşturulan sanal cihazlarına aynı otomasyon testi uygulanmıştır. Elde edilen veriler üzerinde sanal cihazlarda test ve gerçek cihazlarda testlerin karşılaştırılması yapılmıştır.

1.1. Yazılım Geliştirme Yaşam Döngüsü

Yazılım Geliştirme Yaşam Döngüsü (Software Development Life Cycle, SDLC), bir yazılım henüz fikir aşamasındayken başlayarak son kullanıcının (müşterilerin) kullandığı zamanı da kapsayan geniş bir zaman diliminde, yazılımda yapılan geliştirme adımlarının tamamına denir. Bir yazılımın gelişim süreci, o yazılımı daha planlı bir şekilde geliştirmek için zamana ve içeriklere göre belirli aşamalara bölünmüştür. Yazılım testi ise bu döngünün en önemli aşamalarından birisidir (Nalbant, 2020).

Yazılım Geliştirme Yaşam Döngüsü sarmal hareketi olan bir döngüdür. Bu döngü adımları planlama, tasarım, geliştirme, entegrasyon ve testler, uygulama, bakım ve değerlendirme olmak üzere yedi adımdan oluşur (Şeker, 2015). Bu adımlar çoğu kaynakta ortak adımları içerse de bazı kaynaklar adım isimlerini değiştirmiş, bazı adımlar eklenerek on adım haline getirmiş, bazı kaynaklar ise adımları çıkararak veya birleştirerek 4 adıma kadar indirgemiştir. Yazılım Geliştirme Yaşam Döngüsünün temel adımları diyagramı Şekil 1.'de verilmektedir.

Şekil 1. Yazılım Geliştirme Yaşam Döngüsü temel adımları diyagramı (Şeker, 2015).

Yazılım Geliştirme Yaşam Döngüsünün ilk aşaması olan planlama aşamasında proje lideri proje şartlarını değerlendirmektedir. Yazılım nasıl kullanılacak, yazılımın girdisi olarak kullanılacak veriler nelerdir, yazılım çıktısından beklentiler nelerdir ve yazılımın hedef kitlesi kimlerdir soruları yanıtlanarak yazılım projesi için bir yol haritasının çıkarıldığı adımdır (Hancı, 2017).

Tanımlama aşaması; yapılması düşünülen yazılım projesi için ilk fikirlerin belirlenmesinin yanı sıra, yeni bir yazılım için gereken tüm detayların toplanmasını içermektedir. Son kullanıcıların ihtiyaçlarını belirlemek için araştırma ve analiz yapılır (Gökgöz ve Keskinkılıç, 2018).

Tasarım aşaması, yazılımın iskeletinin nasıl olacağına karar verilir. Proje içerisinde yapılacak işlemler adım adım belirlenir ve proje planı oluşturulur. Yapılacak proje bir uygulama ise, uygulamanın açılış ekranı, logosu, uygulama içi menülerin içeriği, menü konumları, ana renkler vb. uygulamanın ana unsurlarını tasarlama işlemleri yapılır (Chauhan ve Saxena, 2013).

Geliştirme adımı, geliştiricilerin artık kod yazdığı ve uygulamayı önceki tasarım belgelerine ve ana hatlarıyla belirtilen özelliklere göre oluşturduğu kısımdır. Ürün program

kodu, tasarım özelliklerine göre oluşturulmaktadır. Teorik olarak, önceki tüm planlama ve ana hatlarıyla gerçek geliştirme aşamasını nispeten basit hale getirmelidir (Şeker, 2015).

Entegrasyon ve Testler aşamasında artık yazılımda herhangi bir hata olmadığından ve son kullanıcı deneyiminin hiçbir noktada olumsuz etkilenmeyeceğinden emin olmak için test edilmesi gerekmektedir. Testten sonra, yazılımın genel tasarımı bir araya getirilir. Farklı modüller veya tasarımlar, geliştirici çabaları yoluyla, genellikle daha fazla hata veya kusuru tespit etmek için eğitim ortamlarından yararlanarak birincil kaynak koduna dahil edilir (Şeker, 2015).

Uygulama aşamasında, test aşamasından geçen uygulama pazar yerine hazır bir şekilde paydaşlara teslim edilir ve son kullanıcının kullanımına sunulur. Son kullanıcı yazılımı kullanırken hata çıkıp çıkmadığı takip edilir. Olası hata çıkma durumlarında yeni fikirler öne sürülür (Şeker, 2015).

Yazılım pazara ulaştığında Yazılım Geliştirme Yaşam Döngüsü sona ermez, son aşama olan bakım aşamasına geçilir. Geliştiriciler artık bir bakım moduna geçmeli ve son kullanıcılar tarafından bildirilen sorunları ele almak için gereken tüm etkinlikleri uygulamaya başlamalıdırlar. Ayrıca, dağıtımdan sonra yazılımın ihtiyaç duyabileceği değişiklikleri uygulamaktan geliştiriciler sorumludurlar (Şeker, 2015).

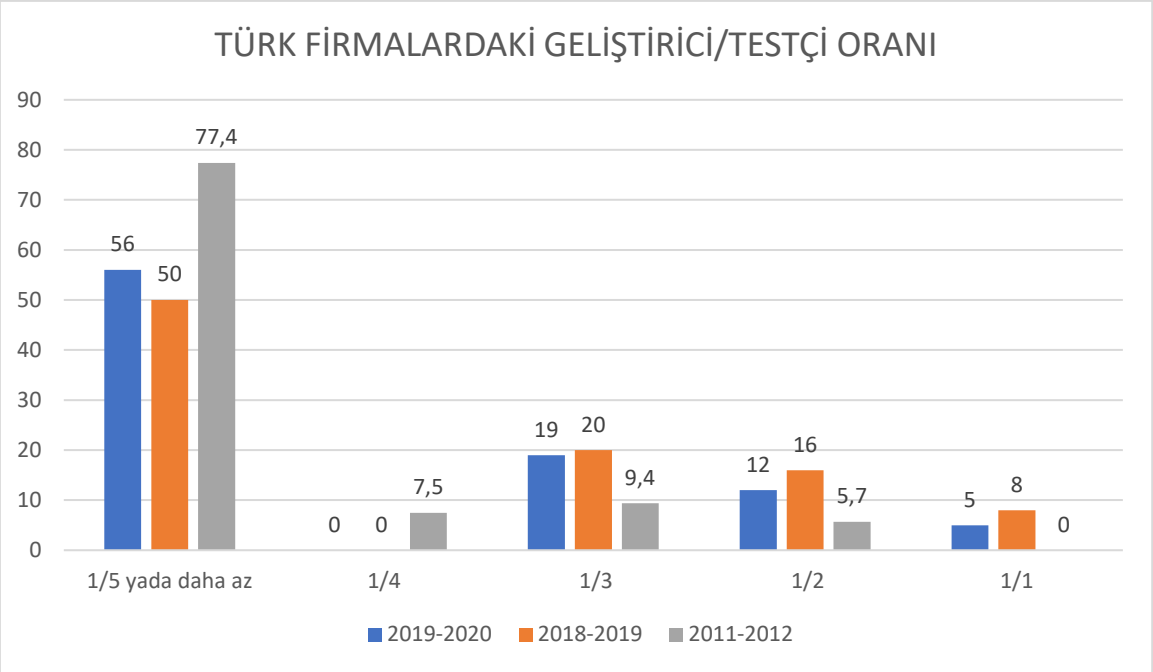
1.2. Yazılım Testi Nedir, Neden Kullanılır?

Yener vd.'ne (2019) göre bir yazılım projesinde istenilen özellikleri doğru bir şekilde yansıtmak, yazılım geliştiricisinin en önemli vazifesidir, aslında bu yazılımın kalitesini belirlemektedir. Yazılım geliştirici her ne kadar doğru ve güzel bir yazılım yazmış olsa dahi her yazılımda mutlaka hatalar bulunmaktadır. Bu hataların tespit edilmeye çalışılması ve yazılıma bir başka gözle bakılması işlemine yazılım testi denir. Yazılım testi, ilgilenilen bir veya daha fazla özelliği değerlendirmek için manuel veya otomatik araçlar kullanılarak yazılım ögelerinin yürütülmesini içerir. Yazılım testi basit anlamda yazılımda hata bulmak amacıyla yazılımın çalıştırılmasıdır. Yazılım testi, yazılım uygulamalarının risklerini anlamak için yazılımı bağımsız ve nesnel olarak incelemektir. Yazılımdaki kalitenin

istenildiği gibi gerçekleştirilebilmesi için, yazılımdaki gereksinimlerin doğru anlaşılması ve yazılım testlerinin olabildiğince hatasız yapılabilmesi gerekmektedir (Yener vd., 2019).

Yazılım testi; sistem gereksinimlerini karışılıyor mu, sistem beklenildiği gibi çalışıyor mu, kullanıcıların ihtiyaçlarını karşılıyor mu, beklentileri tatmin ediyor mu, yazılımda hata var mı gibi soruları yanıtlamak ve ürünün kalitesini göstermek amacı ile yapılmaktadır.

Firmalar arası rekabet, yazılım testine verilen önemi geçtiğimiz yıllara kıyasla arttırmıştır. Sensor Tower şirketinin yapmış olduğu araştırmalara (2022) göre Türkiye'de yazılım firmalarının çalıştırdığı yazılım geliştirici ve yazılım test mühendisi oranı artma eğilimine girmiştir. Özellikle 2018-2019 yılları arasında yazılım firmalarının çalışan oranlarına bakıldığında yazılım geliştirici ve yazılım test mühendisi sayısı eşit olan şirketler bütün şirketlerin %8'ini oluşturmaktadır. Bu oran 2019-2020 döneminde %5'e düşmüştür ancak 2011-2012 senesine bakılınca aynı sayıda çalışan çalıştıran şirket sayısı 0 olduğu için artış yazılım test mühendislerinin piyasada daha fazla iş bulduklarını ve şirketlerin daha fazla önem gösterdiklerini anlamaktayız. 2011-2012 yılında yazılım geliştirme ekibi test ekibinden 5 kat daha büyük olan şirketler piyasadaki şirketlerin %77'sini oluşturmaktaydı ancak 2020 de bu oran %56'ya düşmüştür (Sensor Tower, 2022). Bu verilerin tamamı Şekil 2.'de gösterilmektedir.



Şekil 2. 2012, 2019 ve 2020 yıllarındaki geliştirici ve test alanında çalışan işçi oranları (“Turkish Testing Board,” 2022.).

Yazılım testi yapılırken gereksinimler ve senaryolar doğru belirlenmelidir. Test işlemlerinde artık hata bulunamıyor olması o yazılımı hatasız yapmaz, uygulanan teknikler ve oluşturulan senaryo için hata olmadığı anlamına gelir. Hatasız yazılım yazmak nasıl imkânsızsa, hiçbir noktası atlanmamış bir test yapmak da aynı şekilde imkânsızdır. Test çalışması sadece yazılımın ilk yazıldığı zaman uygulanan bir çalışma değil, yazılımın yaşam döngüsü içinde sürekli uygulanması gereken bir çalışmadır. Yazılımda yapılan iyileştirilmeler, eklenen özellikler vb. güncellemelerin hepsinde uygulanması gereken bir çalışma olmalıdır (“What is Software Testing and How Does it Work? | IBM,” 2022).

1.3. Yazılım Testi Yaşam Döngüsü

Yazılım Testi Yaşam Döngüsü (Software Testing Life Cycle, STLC); yazılımın sunmak istediği özellikleri karşılama derecesini öğrenmek için yapılması istenen test adımlarından oluşur. Günümüzde yazılımların kalitesine verilen değerin artmasıyla birçok firma için Yazılım Geliştirme Yaşam Döngüsünün en önemli adımların biri olmuştur. Sanılanın aksine yazılım testi sadece tek bir adımdan oluşan bir aktivite değildir. Yazılım Testi Yaşam Döngüsü farklı kaynaklarda, farklı sayıda ve isimlerde adımlara sahip olsa da

temelde hepsi aynı şeyleri ifade etmektedir. Yazılım Testi Yaşam Döngüsü, altı adımdan oluşmaktadır. Bu adımlar sırasıyla; ihtiyaç analizi, test planlaması, test senaryosu geliştirme, test ortamı kurulumu, test uygulaması ve uygulama sonrasıdır (Hamilton, 2022a).

Yazılım Test Yaşam Döngüsünün ilk adımı olan ihtiyaç analizi adımında test ekibi, test edilebilir gereksinimleri belirlemek için gereksinimleri bir test bakış açısından inceler. Kalite güvencesi ekibi gereksinimleri tam olarak belirlemek için paydaşlarla iletişime geçer. Gerçekleştirilecek test türü bu aşamada belirlenir ve test öncelikleri saptanır (Hamilton, 2022a).

Test planlaması adımında kalite güvencesi ekibi, test için ayrılmış bütçe ve zaman tahminleriyle birlikte test stratejisini belirler. Test taslağı çıkarılır, test aracı seçilir, kaynak planlanması ve görev dağılımı yapılır (Hamilton, 2022a).

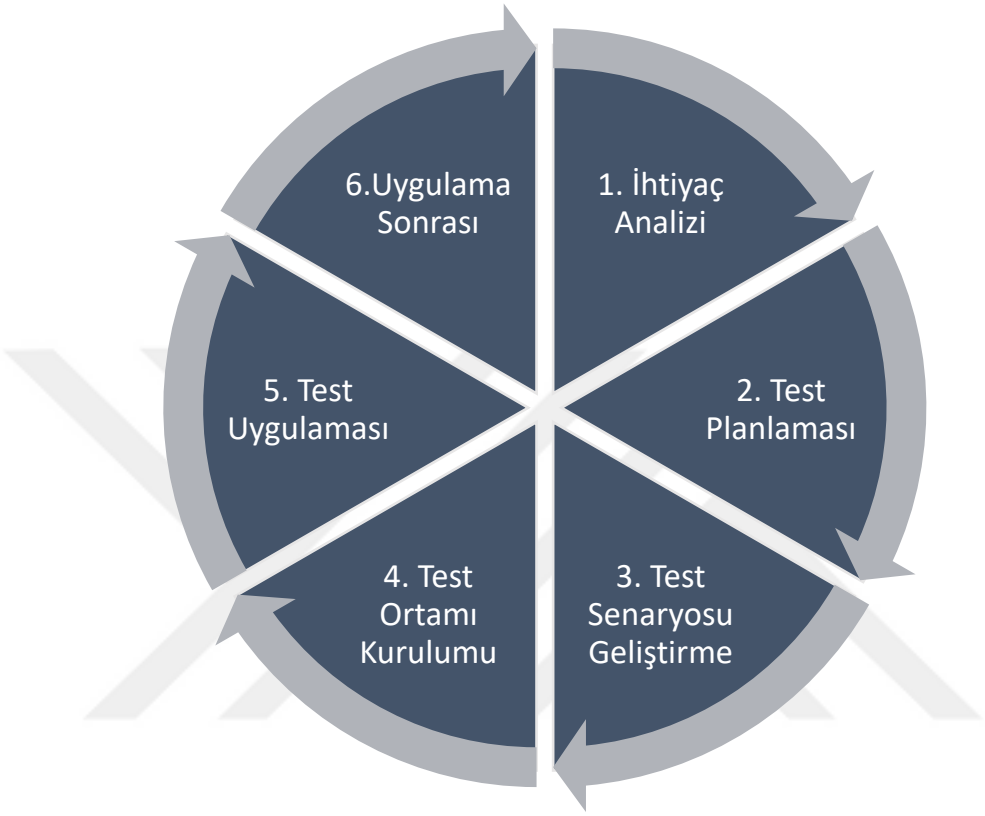
Test senaryosu geliştirme adımı, test planı hazır olduktan sonra test senaryolarının, komut dosyalarının oluşturulmasının ve işlenmesinin yapıldığı adımdır (Hamilton, 2022a).

Test ortamı kurulumu adımında, yazılımın test edilmesi için gereken yazılımlara ve donanımlara karar verilir. Yazılım geliştirme ekibi test ortamını sağlıyorsa test ekibinin bu aşamaya katılmasına gerek yoktur. Duman testi bu aşamada yapılır. Yapı doğrulama testi veya yapı kabul testi olarak da adlandırılan duman testi, bir programın en önemli işlevlerinin çalıştığını doğrulayan ancak daha ince ayrıntılara girmeyen kapsamlı olmayan bir yazılım analizidir (Hamilton, 2022a).

Test uygulaması adımı, önceki adımlarda hazırlanan test planı ve senaryoların test ekibi tarafından uygulandığı Yazılım Testi Yaşam Döngüsünün ana adımıdır. Test senaryoları yürütüldükten sonra ortaya çıkan hatalar bu adımda raporlanır. Raporlanan hatalar yazılım geliştirme ekibi tarafından düzeltildikten sonra senaryo yeniden yürütülür ve bu döngü hata çıktısı alınmayana kadar devam eder (Hamilton, 2022a).

Uygulama sonrası adımı ise Yazılım Testi Yaşam Döngüsünün son adımıdır. Bu adımda test kapanış raporlaması, test tamamlama matrislerinin toplanması ve sonuç çıktılarının alınması gibi işlemler bu adımda yürütülür. Test ekibi yapılan bu testten sonra;

testin kapsamı, maliyeti, süresi gibi temel unsurları kendi arasında değerlendirir. Sonuç çıktıları ve diğer raporlar ürün kalite raporuyla birlikte müşteriye sunulur (Hamilton, 2022a). Yazılım testi yaşam döngüsünü gösteren diyagram Şekil 3.'de verilmektedir.



Şekil 3. Yazılım Testi Yaşam Döngüsü diyagramı

1.4. Yazılım Test Seviyeleri

Uluslararası Yazılım Test ve Kalite Derneğinin (ISTQA) yayınladığı test uzmanlığı dersleri kaynağında, yazılım test seviyeleri dört ayrı başlıkta (birim testleri, entegrasyon testleri, sistem testleri ve kabul testleri) gruplanmıştır. Gruplama; testlerin yapılırken yazılım geliştirme yaşam döngüsünde yazılımın bulunduğu aşamaya göre yapılmıştır ("Turkish Testing Board," 2022).

Birim testi, programın tasarım spesifikasyonunda belirtilen gereksinimleri karşıladığını göstermesi gereken, bir laboratuvar ortamında geliştirici tarafından yürütülen bir test olarak yani bir uygulamanın birim adı verilen test edilebilir en küçük parçalarının

düzgün çalışması için birbirinden bağımsız olarak test edilmesi işlemine denir (Koomen ve Pol, 1999). Bu test metodolojisi bazen yazılım geliştirme aşaması sırasında test mühendisleri haricinde yazılım geliştirici tarafından da yapılabilir. Bu testin amacı her bir birimin kodlarının diğerlerinden izole olarak istenildiği gibi çalışıp çalışmadığını kontrol etmektir.

Watkins ve Mills'e (2010) göre entegrasyon testi, yazılım modüllerinin mantıksal olarak entegre edildiği ve grup olarak test edildiği bir test türü olarak tanımlanır. Tipik bir yazılım projesi, farklı programcılar tarafından kodlanmış birden çok yazılım modülünden oluşmaktadır. Bu test seviyesinin amacı; modüllerin birbirine birleştirildiğinde bu yazılım modülleri arasındaki etkileşimdeki kusurları ortaya çıkarmaktır (Watkins ve Mills, 2010).

Sistem testi, sistemin bir bütün olarak test edilmesi anlamına gelir. Sistemin beklendiği gibi çalışıp çalışmadığını doğrulamak için tüm modüller bütünleştirilmektedir (Turkish Testing Board, 2022). Sistem testi entegrasyon testinden sonra yapılır. Sistemi bir bütün olarak test etmek için gereksinimler ve beklentiler açık olmalı ve testçinin uygulamanın gerçek zamanlı kullanımını da anlaması gerekir (Watkins ve Mills, 2010).

Kabul testleri; isminden de anlaşıldığı gibi müşterinin teslim almadan önce uygulamanın doğru çalışıp çalışmadığını test etmek için girdiler ve çıktıların uyumluluğunun kontrol edildiği test türleridir (Miller ve Collins, 2001).

1.5. Test Türleri

Yazılımlar test edilirken 3 farkı test türü kullanılır. Bu gruplama testin yapılırken kullandığı yöntemlere göre yapılmıştır. Bunlar beyaz kutu testi, gri kutu testi ve kara kutu testi'dir.

Nouman vd.'ne (2016) göre Beyaz Kutu Testi; girdi ve çıktı akışını doğrulamak için kullanılan bir yazılım test çeşididir. Tasarım, kullanılabilirlik ve güvenliği geliştirmek için yazılımın içyapısının, tasarımının ve kodlamasının test edildiği yazılım test tekniğidir. Beyaz kutu testinde, kod test ediciler tarafından görülebilir, bu nedenle test ekipleri arasında bu test; Statik test, Şeffaf kutu testi, Açık kutu testi, Kod tabanlı test ve Cam kutu testi olarak da adlandırılmaktadır (Nouman vd., 2016).

Beyaz Kutu Testi dahili güvenlik açıkları, kodlama süreçlerinde bozuk veya kötü yapılandırılmış yollar, kod aracılığıyla belirli girdilerin akışı, beklenen çıktı, koşullu döngülerin işlevselliği ve her bir ifadenin, nesnenin ve işlevin bireysel olarak test edilmesini içerir (Nouman vd., 2016).

Beyaz kutu testinin temel hedeflerinden biri, bir uygulama için bir çalışma akışını doğrulamaktır. Beklenen veya istenen çıktılara karşı önceden tanımlanmış bir dizi girdiyi test etmeyi içerir. Böylece belirli bir girdi, beklenen çıktıyla sonuçlanmadığında bir hatayla karşılaşılmaktadır (Neamtiu vd., 2005).

Gri Kutu Testi; bir yazılım ürününü veya uygulamasını, uygulamanın iç yapısı hakkında kısmi bilgi sahibi olarak test etmek için kullanılan bir entegrasyon testi seviyesindeki yazılım test tekniğidir. Gri kutu testinin amacı, hatalı kod yapısı veya uygulamaların uygunsuz kullanımından kaynaklanan kusurları aramak ve belirlemektir. İsminden de anlaşıldığı gibi kara kutu testi ve beyaz kutu testlerinin kombinasyonu gibidir. Gri kutu testi hem kara kutu testinin hem de beyaz kutu testinin birleşik faydalarını sağlaması amacıyla yapılır. Test, tasarımcı bakış açısıyla değil kullanıcı bakış açısı ile yapılır ("What is Grey Box Testing? Techniques, Example," 2022).

Nouman vd. (2016) göre dinamik test olarak da bilinen Kara Kutu Testi ile gereksinim özelliklerine göre her bağımsız bölüm birbirinden ayrı olarak test edilebilir ve kara kutu testinde kodu incelemeye gerek yoktur. Bu test tamamen müşterilerin bakış açısına dayalı olarak yapılır, yalnızca test eden kişi girdi setini ve öngörülebilir çıktıları bilir (Nouman vd., 2016). Tamamen bitmiş ürünler üzerinde de kara kutu testi yapılabilir. Kara kutu testi, yazılım testinde önemli bir rol oynar, sistemin genel işlevsellik doğrulamasına yardımcı olur. Kara kutu testi, müşterilerin gereksinimlerine göre yapılır. Bu nedenle, eksik veya öngörülemeyen gereksinimler kolayca belirlenebilir ve daha sonra ele alınabilir (Beizer, 1995).

Kara kutu testi yapılırken ilk olarak sistemin gereksinimleri ve özellikleri incelenir. Sonra test edilen cihazın bunları doğru bir şekilde işleyip işlemediğini kontrol etmek için geçerli girdiler (pozitif test senaryosu) seçilir. Ayrıca, cihazın bunları algılayabildiğini

doğrulamak için bazı geçersiz girdiler (negatif test senaryosu) de seçilir. Cihaza seçilen girdilerle test senaryoları oluşturulur ve yürütülür. Eğer varsa kusurlar düzeltilir ve yeniden test edilir (Nidhra, 2012).

1.6. Test Yöntemleri

Yazılımlar test edilirken kullanılan 2 farklı yöntem vardır, bunlar manuel test ve otomasyon test. Manuel test; test uzmanının, oluşturulan test senaryolarını otomatik hale getirmeden el ve göz yordamı ile yapmasıdır. Test uzmanı, test işlemini gerçekleştirirken kendini son kullanıcı yerine koyarak ve yazılımdan beklentileri düşünerek uygulama içerisinde işlem yapar. Otomasyon testlerinde görünmesi mümkün olmayan hataları da görebildiği için temel test tekniklerinden birisidir ve öncesinde yapılması zorunludur. Renklerde bulunan hatalar, yazı boyutları, kutucuk konumu sıkıntıları, yazı fontu hataları vb. hataların tespit edilebilmesi için manuel test, test uzmanları tarafından test teknikleri içerisinde en önemli test teknikleri arasına alınır. Ayrıca kullanılabilirlik testi de yalnızca manuel test ile mümkündür.

Otomasyon test, bir test senaryosu paketini yürütmek için tasarlanan otomatik test yazılım araçlarını kullanarak gerçekleştirilen bir yazılım test tekniğidir. Otomasyon testi manuel testin yapamadığı beklenen sonuç ve çıkan sonuçları otomatik karşılaştırır ve bununla ilgili detaylı raporu kendisi çıkarabilir. Yukarıda açıklanan yazılım geliştirme yaşam döngüsünün bir adımı olan bakım adımında uygulanan güncellemelerden sonra yazılımın tekrar test edilmesi gerekir ve otomasyon testinde test kodları saklanarak bu gibi durumlarda tekrar çalıştırılabilmektedir. Bir test paketi otomatikleştirildiğinde insan müdahalesi gerekmez. Otomasyon testi, manuel testleri bitirmek için değil manuel teste harcanan zamanı azaltmak için yapılmaktadır (Dustin vd., 1999).

Tüm iş akışlarının, tüm alanların, tüm olumsuz senaryoların manuel olarak test edilmesi zaman ve para tüketir, otomasyon testi maliyetten tasarruf edilmesini sağlar. Çok dilli uygulamaları veya siteleri manuel olarak test etmek zordur; otomasyon testinde kodlama yapıldığı için site veya uygulama dili önemli değildir. Yazılım testinde otomasyon testi insan müdahalesi gerektirmediğinden, uzun zaman alan testlerde testler günlerce otomatik olarak yürümektedirler. Böylece otomasyon testi test yürütme hızını oldukça

artırmaktadır, manuel testten yaklaşık %70 daha hızlıdır. Sonuçlar güvenilir ve tutarlıdır, manuel test sıkıcı hale gelebilir ve bu nedenle dikkat dağınıklığından kaynaklanan birçok hataya neden olabilir (Berner vd., 2005; Fewster ve Graham, 1999; Kaner, vd., 1993; Rice, 2003).

1.7. Mobil Uygulamalarda Test

Mobil uygulamalar akıllı telefon, tablet ve hatta bilgisayarda çalışmak için tasarlanan yazılımlardır. İlk mobil uygulama 1997 yılında geliştirilmiş "snake" oyunu kabul edilir. Üstünden çok fazla zaman geçmemesine rağmen kısa zamanda hızlı bir gelişim göstererek popüler hale gelmiştir (Takgil, 2016).

Test edilen yazılımların karmaşıklığı gelişen mobil cihaz teknolojileri ile gün geçtikçe artmaktadır. Bu da yazılım testlerinin daha ayrıntılı test edilme koşulunu doğurur. Ballard mobil uygulamaları test ederken içerik, kullanıcı ve uygulama olarak sınıflandırdığı üç ana faktöre dikkat edilmesi gerektiğini belirtmiştir (Ballard, 2007). İçerik, cihazın yerleşik elementleri ile ilgilidir; bataryanın tam şarjlı veya az şarjlı kalma durumu, ekran büyüklüğü, hücresel ağ çekim gücü, Wi-Fi kalitesi vb. Kullanıcı, mobil cihazı kullanan kişinin günlük kullanımdaki hareketlerini ve davranışlarını; uygulama ise yazılan uygulamanın üzerinde çalıştığı cihaz ile birleştirilmesini ifade eder.

Mobil uygulamaların testi diğer uygulama testleri ile temelde aynı özelliklere sahiptirler ancak mobil cihazlardaki yazılım ve donanım farkları mobil uygulamaların test edilmesini zorlaştırır. Günümüzdeki mobil cihazlarda testlerin yapılırken cihazların sahip olduğu ekran boyutu, çözünürlük, Bluetooth, kamera, Wi-Fi gibi bileşenlerin entegrasyonları ve bütün bu donanım farklılıkları göz önüne alınarak, uyumluluklarına dikkat edilerek, test edilmesini gerektirir.

Mobil cihazlarda, özellikle Android cihazlarda, üretilen cihaz çeşitliliğin artması testlerin maliyetini artırmaktadır. International Data Corporation şirketinin yayınladığı verilere göre (2022) 2021 yılında %20'lik pazar payıyla piyasaya liderlik eden Samsung Electronics şirketidir ("Smartphone Shipments Declined in the Fourth Quarter But 2021 Was Still a Growth Year with a 5.7% Increase in Shipments, According to IDC," 2022). Samsung

markası 2021 yılında toplam otuz sekiz farklı yeni ürün çıkarmıştır ve şu anda Samsung'un güncelleme desteği sunulan cihaz sayısı yüzün üzerindedir. Piyasaya yön veren diğer şirketlerden olan Xiaomi, Oppo, Vivo, LG gibi şirketlerin ürettikleri telefonlarla birlikte devasa bir telefon havuzu oluşmaktadır. Bu maliyet artışı test firmalarını uygulama testlerini sanal cihazlar (emülatör) üzerinde yapmaya itmektedir.

1.7.1. Sanal Cihazlarda Test

Emülatörün temelde yaptığı şey gerçekte var olan bir cihazı bilgisayar ortamında simüle etmektir. Böylece her üretilmiş cihazı satın almak gerekmeden o cihazı bilgisayarda aynı özelliklerde kullanım imkânı sağlar. Emülatör, gerçek bir cihazın neredeyse tüm özelliklerini sağlamaktadır; gelen telefon aramalarını ve metin mesajlarını simüle edebilir, cihaza konum bilgisi atanabilir, farklı ağ hızlarını simüle edebilir, cihaz döndürme işlemini simüle edebilir ve diğer donanım sensörlerini simüle edebilir ("Run apps on the Android Emulator," 2022).

Emülatörleri test işlemlerinde kullanmanın birçok avantajı vardır. Bu avantajlar aşağıda maddelerle sıralanmıştır.

- Sanal cihazlar için üretilen cihaz sayısının bir önemi yoktur, bir emülatör bütün üretilmiş gerçek cihazları simüle edebilir.
- Birden fazla cihazı taşımak gibi bir sorun oluşmaz, hepsi bilgisayarın içerisindedir.
- Emülatörler makine düzeyinde montaj dillerinde yazılmıştır.
- Hata ayıklama söz konusu olduğunda emülatörler daha başarılıdır.
- Emülatörler internetten çok kolay bir şekilde indirilerek ulaşılabilirler.
- Bağlantı sorunları çıkarmaz.
- Kablo kalabalığı ve karışıklığı çıkmaz.
- Isınma problemi yaşamaz, bataryası bitmez.
- Fiziksel bir donanımı olmadığı için asla donanım arızası veremezler.

Emülatörler her ne kadar gerçek cihazın yapabildiği birçok şeyi yapsa da tabi ki bir gerçek cihazın yerini tutmayacaktır. Aşağıda yazılım testinde sanal cihazların kullanılmasının dezavantajları maddelerle sıralanmıştır.

- Gerçek cihazın elle kullanımda anlaşılabilecek ışık akısı, renk görünürlüğü, yürürken kullanım, tek elle kullanım gibi birçok detay simülatörlerde anlaşılamaz.
- Her ne kadar birebir aynı cihaz simüle edilse de cihaz içerisindeki bileşenler ve bilgisayar ortamındaki bileşenler aynı değildir.
- Bir sanal cihazda az şarjlı bir test gerçekleştirmeniz mümkün olmayacaktır veya uzun süreler süren performans testlerini de tam verimli şekilde yapamayacaksınız.
- Yazılımın ne kadar şarj yediği konusunda sanal cihazlar herhangi bir geri dönüş yapamazlar.
- Sanal cihazlar uygulama kullanılırken gelen aramalar ve kısa mesajlarda veri kısma veya gerçekleşen kesintileri test uzmanına gösteremez.
- Her ne kadar emülatörler gelişmiş olsa da cihaz performansları zaman zaman orijinal cihazlara göre daha yavaş olma eğilimindedirler.

1.7.2. Gerçek Cihazlarda Test

Gerçek cihaz; fiziksel donanımlarla, sanal olmayan bir ortamda üretilmiş bütün cihazları kapsamaktadır. Test edilecek yazılım hangi cihaz hedeflenerek yazılıyorsa o cihazın temin edilmiş ve test yapılacak makineye bir kablo aracılığı ile bağlanmış olması gerekmektedir.

İKİNCİ BÖLÜM
ÖNCEKİ ÇALIŞMALAR

Bu çalışmanın ana konusu olan otomasyon ile mobil yazılımların testlerinin fiziksel ve sanal cihazlar üzerinde gerçekleştirilmesi üzerine literatürde çok fazla çalışma bulunmamaktadır.

Kaner vd. (1993), bilgisayar yazılımlarının gerçek dünya koşullarında nasıl test edebileceğini açıklayan öncü bir çalışma yapmışlardır.

Koomen ve Pol (1999), yazdıkları kitaplarında mevcut test süreçlerini iyileştirmek için önerilerde bulunmuş ve Test Process Improvement modelini açıklamışlardır. Geliştirdikleri bu modelle, işletmelerin üretecekleri projeleri en kaliteli, en düşük maliyet, en kısa teslim süresi ile tamamlanmasını hedeflemişlerdir.

Nidhra (2012), kara kutu ve beyaz kutu test teknikleri ile ilgili literatür çalışması yapılmış ve bir beyaz kutu testi çalışması yapılmıştır. Bu çalışma, test alanında kara kutu test teknikleri için güzel bir kaynak olmuştur.

Gao vd. (2014), mobil uygulama testleri ve otomasyon testleri üzerine çalışma yapmışlardır. Mobil uygulamalarda test süreçleri hakkında bilgi veren bu çalışma aynı zamanda farklı test uygulamalarında hangi otomasyon testlerin yapılabileceğini göstermiş ve karşılaştırmışlardır.

Gunasekaran ve Bargavi (2015) çalışmalarında, Android ve iOS için üretilen 5 otomasyon test aracını kıyaslamış ve verimliliklerini araştırmıştır.

Nouman vd. (2016), yazılım testindeki son eğilimleri araştırmış ve bu alandaki farklı ticari araçların bir analizi de sunmuşlardır. Ticari alanda faaliyet gösteren bazı test araçlarının listesini yapmış ve hangi otomasyon testlerini yapabildiklerini listelemiştir.

Takgil (2016), Android işletim sistemli mobil cihaz uygulamaları için yazılım testinde yaşanan başlıca zorluklar incelenmiş ve bu zorluklar için otomasyon test çözüm önerisi yapılmıştır.

Koziokas vd. (2017), web ve hibrit uygulama modellerinde kullanılabilirlik testi yaparak karşılaştırmıştır. Gerçek kullanıcılar üzerinde yapılan deneye göre ticari faaliyet gösteren firmaların web sayfaları yerine hibrit mobil uygulamaları kullanıcılar tarafından daha da beğeniliyor ve daha kullanışlı bulunuyor. Bu çalışma mobil uygulamanın ticari faaliyetler için önemini deneysel olarak kanıtlamıştır.

Sneha ve Malle (2017), bazı önemli ve yüksek talep gören yazılımlar tartışılmış ayrıca farklı platformlarda kullanılan çeşitli otomatik test araçları üzerine bir çalışma sunulmuştur. Otomasyon testinin manuel teste göre önemi vurgulanmıştır.

Hancı (2017), yazılım testi tasarım tekniklerinin matematiksel modellemesini ve bu modellemenin analizini çıkaran bir çalışma yapmıştır. Gereksinim, yapı, deneyim, statik bazlı test tekniklerini matematiksel karşılaştırarak en iyi test tekniklerini tespit etmiştir.

Yener vd. (2019), test otomasyon geliştirmenin önemi üzerine çalışmışlardır. Manuel test ve otomasyon test arasında verimlilik karşılaştırması yaparak otomasyon test kullanımının önemini göstermişlerdir.

Zhan ve Yan (2019), yazılım şirketinin, mobil uygulama testi alanını araştırmışlardır. Ağırlıklı olarak işletmelerde popüler olan mobil uygulama test yöntemlerini ve araçlarını incelemişlerdir. Ayrıca yazılım şirketlerinde mobil uygulama testleri ile ilgili karşılaşılan zorlukları ve o zorlukları şirketlerin nasıl çözdüklerini incelemişlerdir.

Kaymak (2020), yazılım test araçlarını karşılaştırmışlardır. Ayrıca manuel testlerin yerine otomasyon testlerinin kullanılması gerektiğini savunmuşlardır.

Soyer (2022), yazılım testlerindeki güncel standartları gösteren bir çalışma yapmıştır. Yazılım testlerinde kullanılan teknikleri incelemiş ve uluslararası standartlarla (ISO vb.) bağlantılarını kurmuştur.

ÜÇÜNCÜ BÖLÜM
MATERYAL YÖNTEM

Yazılan otomasyon testi uygulamayı sıfırdan başlatıp gerekli bölüm seçimine kadar gelmektedir. Bu aşamalarda ayrı ayrı yazılmış birimlerin birleşimini test ettiği için uygulanan test seviyesi entegrasyon testi olarak gerçekleştirilmiştir. Uygulama test yöntemi için otomasyon kodları yazılarak, otomasyon test yöntemi gerçekleştirilmiştir. Test için seçilen uygulama halihazırda piyasaya sürülmüş bir uygulama olduğu için yazılım kodlarına erişilememiş ve test türü olarak siyah kutu testi uygulanmıştır. Uygulamanın donanım ve yazılımsal açısından performansını kontrol etmek için performans testi; uygulamanın hata vereceği veya çökeceği sınırları belirlemek için stres testi; uygulamayı kullanıcı bakış açısıyla, hissini ve kullanıcı dostu olup olmadığını belirlemek için kullanılabilirlik testi; farklı cihazlarda, donanımlarda, işletim sistemlerinde uyumluluğu görmek için uyumluluk testi olmak üzere 4 adet fonksiyonel olmayan test çeşitleri kullanılmıştır.

- Performans testi, sistemin takıldığı ve hata verdiği noktaları bulmak için yük altında test edilmesine denir. Ana hedef, donanım ve yazılım açısından ölçeklenebilirlik, kullanılabilirlik ve performansı test etmektir. CPU kullanımı, bellek kullanımı, önbellek tutarlılığı, veri tutarlılığı, güç tüketimi, ağ bant genişliği kullanımı gibi kaynak yönleri de performans testinin bir parçası olarak izlenir ve raporlanır. Performans analizi, ürün geliştirmenin her aşamasında uygulanmalıdır (Sarojadevi, 2011).
- Stres Testi (Stress Testing), yazılım uygulamasının kararlılığını ve güvenilirliğini doğrulayan bir tür yazılım testidir. Stres testinin amacı, yazılımın son derece ağır yük koşulları altında sağlamlığını ve hata işleme yeteneklerini ölçmek ve yazılımın zorlu durumlarda çökme durumunu test etmektir. Bu test normal çalışma noktalarının ötesinde testler yaparak ve yazılımın aşırı koşullar altında nasıl çalıştığını değerlendirir (Chan, 2004). Kısacası Stres Testinin amacı donanımın ya da yazılımın hata vereceği ve çökeceği sınırları belirlemektir. Ayrıca, sistemin aşırı koşullar altında etkin hata yönetimi gösterip göstermediğini de kontrol eder. Örneğin Windows bilgisayarlarda hali hazırda bulunan Not Defteri uygulamasına 5 GB'lık bir

veri kopyalanıp yapıştırıldığında Not Defteri yanıt vermeyecek ve çökecektir (Hamilton, 2022b).

- Kullanılabilirlik Testi (Usability Testing), bir uygulamayı görünüşlerini, kullanıcı hislerini ve kullanıcı dostu olup olmadığını kontrol etmek için son kullanıcı bakış açısıyla yapılan testlere denir. Keşif testi, çapraz tarayıcı testi ve erişilebilirlik testi olmak üzere üç adet kullanılabilirlik testi vardır (Koziokas vd., 2017).
- Uyumluluk Testi (Compatibility Testing), oluşturulan uygulamanın veya yazılım ürününün; donanım, işletim sistemi, veritabanı veya diğer sistem yazılımlarıyla uyumlu olup olmadığını test eder (Yoon vd., 2008)

Bu tez çalışması yapılırken gerçek cihazlar alınmış ve karşılaştırma yapılması için Android Studio ve Genymotion kullanılarak satın alınan cihazların sanal emulatorleri oluşturulmuştur. Yazılım testinin otomasyonunu cihaza aktarmak için seçilen uygulama Appium Server GUI, otomasyonun kodlanması için de Java dili kullanan IntelliJ IDEA yazılımı kullanılmıştır.

3.1. Android Studio

Android mobil cihazlar için Linux temelli yazılmış, açık kaynak kodlu işletim sistemi markasıdır. Android ilk defa 2003 yılının Ekim ayında hayatımıza Kaliforniya'da dört ortakla kurulmuş bir şirkettir. Bu şirket 2005 yılında Google Inc. tarafından satın alınmıştır. Satın alındıktan sonra Open Handset Alliance (OHA) birliğini kurmuştur. Birlik mobil cihazlar için kaynakların açık olma standartlarına bağlanmak için kurulmuştur. Bu birliğin şu an 84 üye firması bulunmaktadır. 2008 yılında ilk sürümü olan Android 1.0 yayınlanmış ve o günden bugüne kadar hızlı bir büyüme göstererek dünyanın en çok kullanılan işletim sistemi olmuştur. 2022 yılında aktif kullanıcı sayısı 2,5 milyarı geçmiştir ("Set up for Android Development ," 2022).

Android Studio, Android'in üretmiş olduğu Java temelli bir resmi tümleşik geliştirme ortamı (IDE) yazılımıdır. Android Studio kullanılarak fiziksel cihazlara geliştirme yapılabilir ve sanal cihazlar oluşturulabilir. Bu çalışmada sanal cihazları oluşturmak için kullanılmıştır.

3.2. Genymotion

Genymotion, 2011 yılında Arnaud Dupuis tarafından kurulan ve şu anda Paris, Lyon ve San Francisco'da şubeleri bulunan çevrimiçi ve çevrimdışı kullanılabilen Android platformudur. Bu uygulama ile bilgisayarda çevrimdışı bir emülatör oluşturabilir veya çevrimiçi bir şekilde gerçek cihaz kütüphanesine erişerek gerçek cihazlarda uygulama testleri yapmaya olanak sağlar. Çevrimiçi çalışma yöntemi bir bulut sistemi üzerinden yapılır. Gerçek cihazlar bulut sisteminde var olan bilgisayarlarla bağlanmış ve test için hazır hale getirilmiştir. Ancak bu tez çalışmasında sanal cihaz üretimi için kullanılmıştır dolayısıyla bulut sistemi kullanılmamıştır ("Android Emulator for app testing Cross-platform Android Emulator for manual and automated app testing," 2022).

3.3. Appium

Appium piyasada en çok kullanılan otomasyon araçlarından birisidir. Appium, iOS mobil, Android mobil ve Windows masaüstü platformlarında yerel, mobil web ve hibrit uygulamaları otomatikleştirmek için açık kaynaklı bir araçtır. Python, Java, PHP, JavaScript vb. birçok programlama dilinin kullanıma izin veren bir yazılımdır. Appium platformunun avantajları aşağıda maddeler halinde listelenmiştir.

- Appium, aynı API'yi kullanarak farklı platformlara (iOS, Android vb.) testler yazılmasına izin verir.
- Hem sanal cihazlarda hem de gerçek cihazlarda test yapılmasına imkân tanır.
- Appium, test uzmanlarının kodu sık sık değiştirmeden veya yeniden derlemeden hem yerel hem de platformlar arası mobil uygulamaları değerlendirmesini sağlar.
- Appium ayrıca sağlam bir kayıt ve oynatma aracı sağlar. Appium ile çalışırken, testçiler kayıt ve oynatma işlevi aracılığıyla test çalışmalarını hızlandırmak için Inspector'ı kullanabilir.
- Appium herhangi bir test çerçevesi ile uyumludur. Appium, mobilite ekiplerine seçtikleri test çerçevesini kullanma esnekliği sağlar. Eskiden testler yalnızca Javascript üzerinden Apple için UI Otomasyonu kitaplığı

kullanılarak yazılarak veya Java tabanlı testler yalnızca UI Automator of Google üzerinden yazılabilirdi.

- Bulut tabanlı testi destekler, Sauce Labs, BrowserStack ve Testdroid gibi servisleri kullanarak test scriptlerinizi bulutta çalıştırabilirsiniz.

Tabi ki avantajlarının yanında dezavantajları da bulunmaktadır. Appium'un dezavantajları da aşağıda maddeler halinde listelenmiştir.

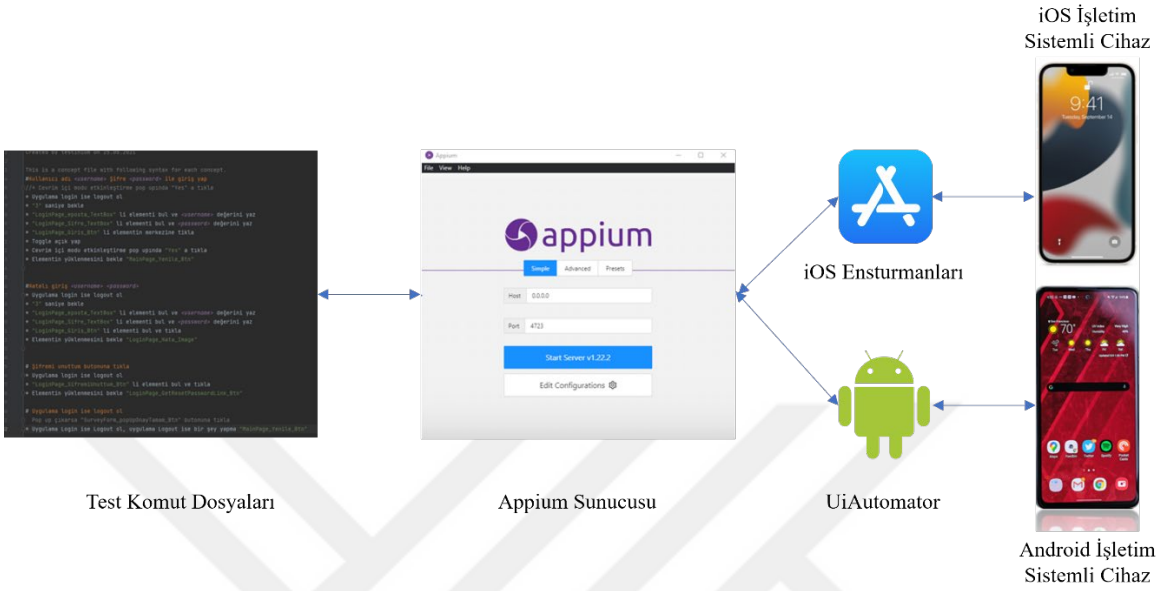
- Appium aynı anda yalnızca bir cihazı çalıştırmanıza izin verir.
- Android 4.1 veya daha düşük sürümler için otomasyon desteği sunmamaktadır.
- Testlerden sonra ayrıntılı bir rapor vermez.
- Appium'da test uzak web sürücüsüne bağlıdır, bu nedenle diğer test araçlarından biraz daha yavaş çalışır.

3.3.1. Appium mimarisi

Appium özünde node.js (ağ bağlantılı uygulamalar için geliştirilen ortam) ile yazılmış bir sunucudur. Sunucu, bir istemci-sunucu mimarisi kullanarak çalışır. İstemci-sunucu mimarisine göre, istemci sunucuda barındırılan herhangi bir hizmetten yararlanmak için sunucuya bağlanır. İstemci ile sunucu arasındaki her türlü iletişim, yanıt ve istek şeklindedir. Appium'da istemci, otomasyonla ilgili istekleri Appium sunucusuna gönderir. Sunucu, kendi yöntemiyle işler ve ardından test sonucuyla yanıt verir. İstemci sunucuya oturum istekleri olarak da bilinen gönderi istekleri gönderir. Bu istekler, bilgileri bir JSON Nesnesi biçiminde taşır ve iletişim, JSON Tel Protokolü kullanılarak yürütülür.

JSON Tel Protokolü, istemci ve sunucu arasında iletişim kurmak için kullanılan mekanizmadır. WebDriver geliştiricileri tarafından geliştirilmiştir. Onlara göre protokol, istemciye bir RESTful API kullanarak maruz kalan bir grup standartlaştırılmış uç noktadır. Bu, WebDriver'ın otomasyon gerçekleştirmek için bir sunucu ve bir istemci ile iletişim kurmasını sağlar. Appium, Selenium JSON Tel Protokolünün bir uzantısı olan mobil JSON Tel Protokolünü kullanır. Bu protokol sadece bir iletişim akışı kurmak için değil aynı

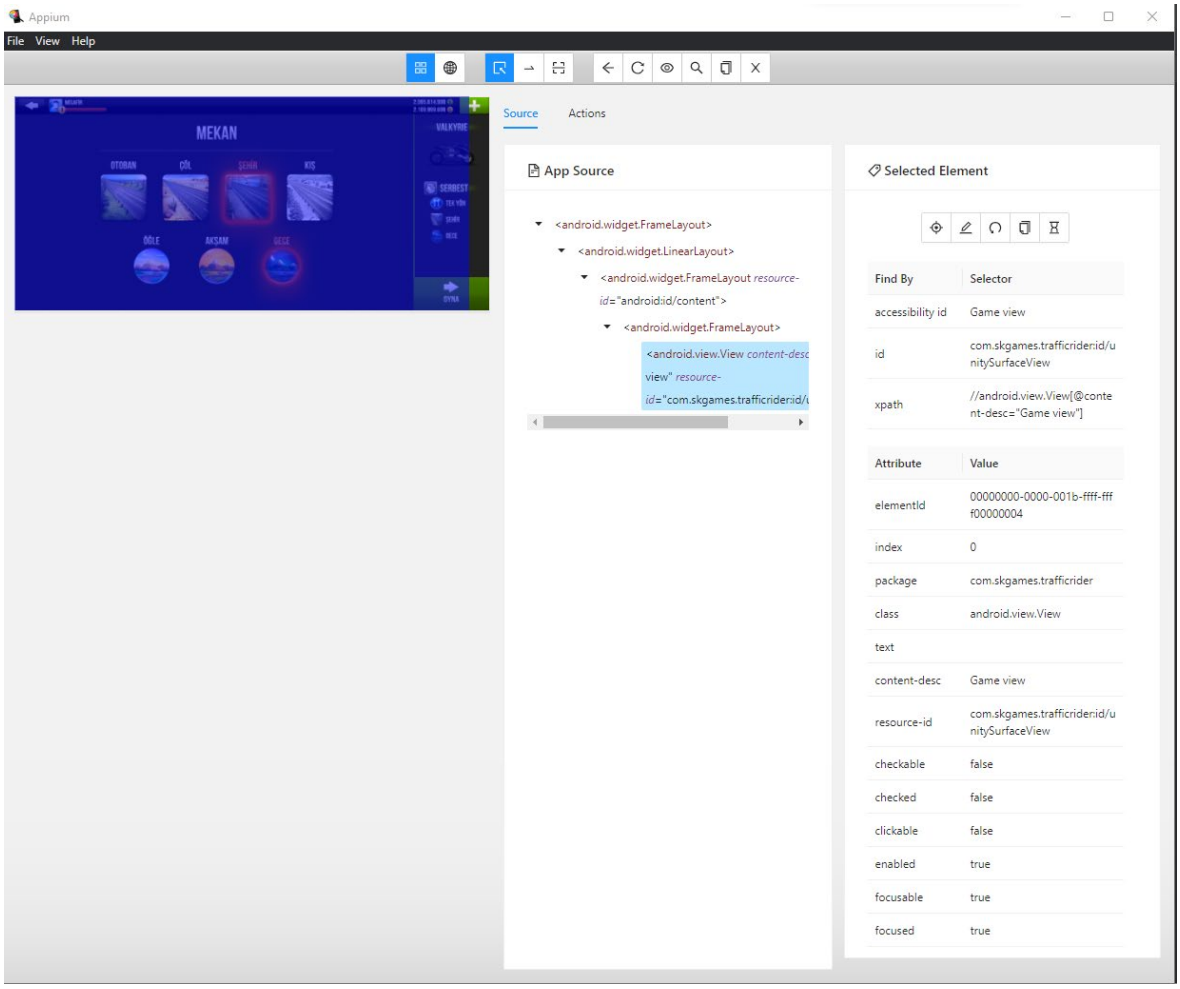
zamanda farklı mobil cihazların davranışlarını kontrol etmek için de kullanılır. Appium mimarisinin çalışma sisteminin basit bir anlatımı Şekil 4.'de verilmektedir.



Şekil 4. Appium Mimarisi Gösterimi

3.3.2. Appium Inspector

Appium Inspector, bir uygulamanın ögelerini aramak ve özel erişim kodlarına erişmemize olanak sağlayan bir Appium uygulamasıdır. Bu uygulama, kod içeriğini bilmediğimiz uygulamaların Java dilinde test otomasyon kodlarını yazarken bize istediğimiz noktanın benzersiz kodunu bulmamızı ve yazdığımız test kodlarının hedefini doğru bir şekilde belirlememizi sağlar. Bu çalışmada kullanılan Appium Inspector'e ait çalışma ekran görüntüsü Şekil 5.'de verilmektedir.



Şekil 5. Appium Inspector Çalışma Ekranına Örnek Bir Görüntü

3.4. IntelliJ IDEA

Java 1995 yılında geliştirilen açık kaynak kodlu bir yazılım dilidir. Java tabanlı bir kod yazımı için bir Java geliştirme ortamına (IDE) ihtiyaç duyulur. Birden fazla Java geliştirme ortamı vardır bunlar; NetBeans, Eclipse, Intellij Idea, BlueJ, JDeveloper, DrJava… 2001'de piyasaya sürülen IntelliJ Idea en çok kullanılan Java geliştirme ortamlarından birisidir. IntelliJ IDEA, diller arası yeniden düzenleme ve veri akışı analizi özelliklerine sahiptir. Bu tez çalışmasında Intellij Idea'nın seçilme sebebi tabi ki sadece bu özelliği değildir. Akıllı tamamlama, zincir tamamlama, diller arası yeniden düzenleme ve veri akışı analizi gibi özellikleri de bu tez çalışmasında seçilmesi için bir etkendir. Ama Intellij Idea'nın büyük proje çalışmalarında yavaşlayabilmektedir, bu özellik bu IDE'nin eksi yönüdür. Bu çalışmada hazırlanan ve uygulanan tez yazılımına ait telefon ekranında

istenilen koordinata basılmasını sağlayan kodlara ait örnek bir görüntü Şekil 6.'da verilmektedir.

Şekil 6. Bu çalışmada hazırlanan ve uygulanan test yazılımına ait örnek bir görüntü

3.5. Vaka Çalışması

Bu bölümde herkesin erişebildiği ücretsiz bir uygulama olan Traffic Rider isimli oyun üzerinden test senaryoları geliştirilmiş ve gerçek cihazlarda ve sanal cihazlarda uygulanmak üzere IntelliJ IDEA ve Appium Server GUI kullanılarak test otomasyonu yazılmıştır. Testler uygulama üzerinde boş yükte ve yüklü şekilde koşulmuş, çıkan sonuçlar raporlanmıştır.

3.5.1. Traffic Rider

Uygulama bir motosiklet sürme oyunudur. Oyun içerisinde trafik arasında ilerleyerek verilen görevleri geçerek bölüm ilerleyebileceğimiz kariyer modu ve trafiksiz alanda serbestçe yol alabileceğimiz bir serbest modu bulunmaktadır. Oyun 3d grafiklere sahip olan basit diye nitelendirilebilecek bir mobil oyundur. Ana sayfasında oyun içi ayarlar için oluşturulmuş bir modül, oyuncu profilinin kaç saat oynadığı, oynanan oyun sayısı, yapılan en yüksek hız, ortalama hız, toplanan para ve diğer bilgiler kullanılarak oluşturulan sürüş notu gibi bilgilerin yer aldığı bir modül, üreticinin diğer oyunlarını sergilediği bir modül, oyunda kullanılan araçların listelendiği ve satın alımların yapıldığı bir modül, oyun içi para ve altınların satın alınabildiği bir modül, reklam izleterek oyun içi para ve altın kazanılabilecek bir modül ve oyun oynamak için oyun tipi seçebileceğimiz bir modül bulunmaktadır.

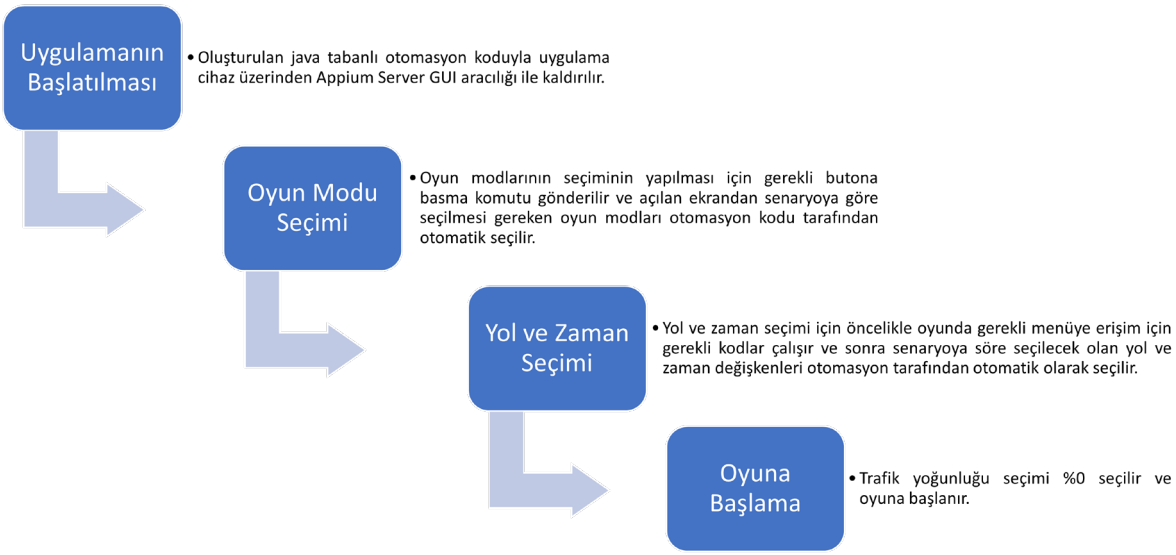
Oyun oynamak için oyun tipi seçilen modülün içerisinde zorluk derecesi bölümlerle arttırılmış bir kariyer modu, kaza yapana kadar bitmeyen bir sonsuz modu, yüz saniye içerisinde en uzun mesafeyi katetmek için oluşturulmuş bir zamana karşı modu ve ister trafikle ister trafiksiz kullanım yapabileceğiniz sonsuza kadar devam edecek olan bir serbest mod bulunmaktadır.

Bu testin amacı olan sanal cihazlarda ve gerçek cihazlarda testlerin farklarını ortaya koyabilmek için oyun tiplerini seçtikten sonra oyunun sanal cihazda oyunun verdiği tepkiler, oyunun akış hızı, kasma veya donma problemleri, renk geçişleri, kalibrasyonu ve uzun süreli oynamalarda cihaz ve oyun tepkileri raporlanarak gerçek cihazda yapılan testlerin raporlarıyla karşılaştırılacaktır. Diğer modüller bu karşılaştırmalar için yeterli verileri veremeyeceğinden sadece oyun modülü test edilmektedir.

Normalde yapılan uygulama testlerinin amacı uygulamaya eklenen özelliklerin istenildiği gibi çalışıp çalışmama durumu görmekken bizim uygulamamız zaten o testlerden geçmiş oluşan hatalar giderilerek son kullanıcının beğenisine sunulmaktadır. Bu tez çalışmasında yapılan testlerin amacı uygulama özelliklerinde hata olup olmadığını sorgulamak değil testlerin koşulduğu cihazların vermiş olduğu sonuçları görmektir.

3.5.2. Test Senaryoları Belirleme

Tüm test senaryolarında uygulama otomasyon ile cihaz üzerinde başlatılarak test süreci başlar. Uygulama başladıktan sonra oyun modu seçimi için gerekli bölüme girilir. Yazılım testinde içinde trafik olmayan oyun modu yani serbest oyun modu seçimi yapılır; yol ve zaman seçimi için gerekli bölüme geçilir. Belirlenen test senaryosuna göre ilgili yol ve zaman seçimleri yapıldıktan sonra trafik yoğunluğunu %0 seçerek oyuna başlanır. Test sürecini anlatan diyagram Şekil 7'de gösterilmektedir.



Şekil 7. Test Adımları Diyagramı

Bir önceki bölümde belirtilen oyun modülünde cihaz performansına etki edebilecek özellikler belirlenmiş ve onlar için test senaryoları yazılmıştır. Traffic Rider isimli oyun için yazılan test senaryoları TRTS kısaltmasıyla yazılmıştır. Örneğin, TRTS05 olarak gösterilen kısaltma 5 numaralı test senaryosunun kısaltmasıdır.

Cihaz performansını etkileyebilecek özellikler oyun içi grafikleri değiştirdiği için seçilmiştir. Bu özellikler; yollar, (otoban, çöl, şehir ve kış) ve saatlerdir (öğle, akşam ve gece).

TRTS01: Yol seçimi otoban ve sürüş saati öğle.
TRTS02: Yol seçimi otoban ve sürüş saati akşam.
TRTS03: Yol seçimi otoban ve sürüş saati gece.
TRTS04: Yol seçimi çöl ve sürüş saati öğle.
TRTS05: Yol seçimi çöl ve sürüş saati akşam.
TRTS06: Yol seçimi çöl ve sürüş saati gece.
TRTS07: Yol seçimi şehir ve sürüş saati öğle.
TRTS08: Yol seçimi şehir ve sürüş saati akşam.
TRTS09: Yol seçimi şehir ve sürüş saati gece.
TRTS10: Yol seçimi kış ve sürüş saati öğle.
TRTS11: Yol seçimi kış ve sürüş saati akşam.
TRTS12: Yol seçimi kış ve sürüş saati gece.

3.5.3. Test Senaryolarını Yazma

Test senaryolarını belirledikten sonra otomasyon kodlarını yazabilmek için Appium Server GUI ve Appium Inspector uygulamasını kullanarak hedef element adreslerini öğrenmek gerekmektedir. Hedef element adresleri için gerçek veya sanal cihazı bağladıktan sonra Appium Server GUI uygulamasından server başlatılır. Inspector uygulamasını kullanarak mobil uygulamayı cihaz üzerinde başlatmak için bazı elementlerin Desired Capability kısmına eklenmesi gerekmektedir. Eklememiz gereken başlıca değişkenler yetenekleri (capabilities); devices name, appPackage, appActivity, UDID, platformName, fullReset ve noReset'dir.

Burada devices name olarak adlandırılan element bizim cihaza verdiğimiz isimdir. Uygulama için bu ismin bir önemi yoktur. İsteğe göre birinci cihaz, iş telefonum, Samsung S10 vb. isimler verilebilir.

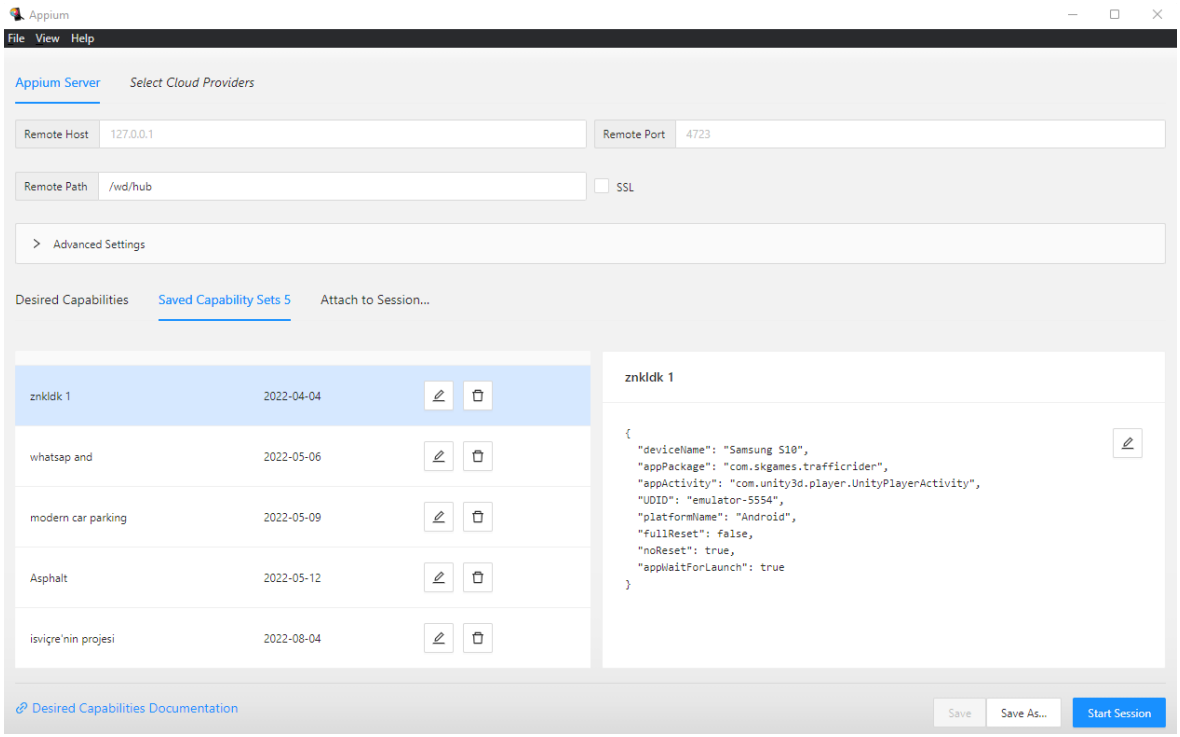
AppPackage elementi ise cihaz üzerinde başlatmak istediğimiz her bir uygulamada bulunan uygulamaya özel (unique) bir değerdir. Test edilmesi istenen uygulamanın Java paketini temsil eder. Test etmek istediğimiz uygulamada bu değeri öğrenmek için uygulama bilgilerini bulmanızı sağlayan üçüncü parti yazılımlar kullanılabilir. Bu testte Kenumir'in yayınladığı apk info isimli uygulaması kullanılmaktadır.

AppActivity elementi, appPackage'ten başlatılması istenen etkinliğin adını temsil etmektedir. Uygulamanın işlemlerini yapabilmesi için açtığı dizinde bulunan "main" dizinini yazmak gerekir. Bu dizin için de appPackage elementini yazmak için kullandığımız uygulama kullanılmıştır.

UDID, bilgisayarda bulunan cihazın özel (unique) değerini ifade etmektedir. Bilgisayarın bağlanan cihaza atadığı bu değeri öğrenmek için bilgisayarın komut istemi uygulamasında "adb devices" kodunu çalıştırarak bağlı bütün cihazların listesini ve UDID kodlarını görebilmekteyiz.

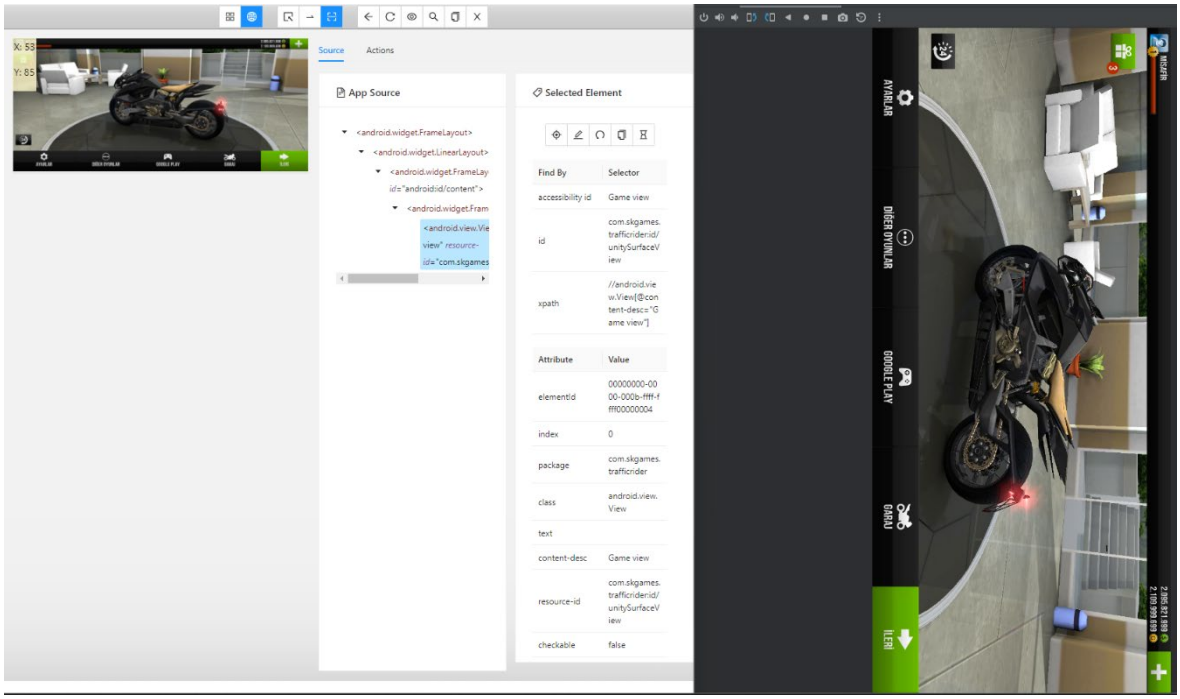
PlatformName, test edilmek istenen cihazın işletim sisteminin yazılacağı elementtir. Bu testte farklı marka telefonlar ve farklı işletim sistemleri kullanılmasına rağmen hepsi Android tabanlıdır bu yüzden bu elemente Android yazılmıştır.

FullReset ve noReset uygulamanın açılırken sıfırlanmasını önbellek verilerinin silinip silinmemesini sağlamak için kullanılan elementlerdir. Yukarıda kullanılan diğer bütün elementler text türü iken bu elementlerin türü boolean'dır. Sadece doğru (true) veya yanlış (false) seçeneği kullanılabilir. Uygulamanın her açılışta giriş bilgilerinin, ilerlemelerinin vb. silinmesi ön belleğin boşaltılması için fullReset elementi true, noReset elementi ise false seçilmelidir. Testler yapılırken her seferinde giriş ekranına erişebilmek için daha önceden yapılan giriş bilgilerinin uygulama önbelleğinden silinmesi gerekir. Ancak bu testte olduğu gibi bazı testlerde ilerlemelerin kaybolması istenmez. Bu testte bütün senaryoların test edilebilmesi için uygulama içerisinde bulunan yol seçimlerinin ve sürüş saatlerinin açık olması gerekmektedir. Uygulama her açılışta önbelleğini temizlerse yapılan ilerlemeler kaybolacağından yol seçimleri ve sürüş saatleri kilitli olacaktır. Bu yüzden bu testte fullReset elementi false, noReset elementi true olarak girilmiştir. Uygulama için yazılan Inspector elementler Şekil 8.'de gösterilmektedir.



Şekil 8. Traffic Rider Uygulaması İçin Yazılan Appium Inspector Elementleri

Uygulama başlatıldıktan sonra Inspector uygulaması üzerinde cihazın anlık ekran görüntüsüne ve uygulamada o an ekranda bulunan her bir noktanın xpath, id, elementId, class, text vb. element değerlerini görüntülenebilmektedir. Bu ekranda test yaparken kullanacağımız noktaların (buton, sekme vb.) benzersiz değerleri bulunmaktadır. Bu elementler otomasyon kodunu yazabilmek için gereklidir. Gerekli olan bütün veriler alındıktan sonra senaryoya uygun bir şekilde kod yazılımı yapılır. Şekil 9.'de Appium Inspector kullanılarak elementlerin tespitinin yapıldığı bir ekran görüntüsü bulunmaktadır.



Şekil 9. Appium Inspector Kullanarak Elementlerin Bulunması

DÖRDÜNCÜ BÖLÜM
ARAŞTIRMA BULGULARI

Bu bölümde, Bölüm 2'de oluşturmuş olduğumuz test senaryolarını önce gerçek cihazlarda sonra da sanal cihazlarda uygulayıp çıkan sonuçların raporlama işlemi yapılacaktır.

4.1. Test Cihazlarına Ait Bilgiler

Yapılacak olan testlerden çıkan sonuçların güvenilir olması için farklı marka ve özelliklerden üç adet cihaz test cihazı olarak seçilmiştir. Bu cihazlar; Xiaomi Mi 8 Lite, Google Pixel 3a, Samsung S10.

Xiaomi Mi 8 Lite isimli telefondan piyasada 2 farklı versiyon bulunmaktadır. Bu cihazlar arasındaki tek fark, bir versiyonda dahili hafıza 64 GB iken diğer versiyonda 128 GB dahili hafıza bulunmaktadır. Bu testte 64 GB'lık versiyon kullanılmıştır. Cihazda 6.26 İnç, IPS LCD teknolojili, 1080x2280 (FHD+) Piksel çözünürlüğe ve 60 Hz ekran yenileme hızına sahip bir ekran bulunmaktadır. 2.2 GHz frekansa sahip 8 çekirdekli Qualcomm Snapdragon 660 (SDM660) işlemciye (chipset) sahip olan cihazda 4 GB da ön bellek (RAM) bulunmaktadır. Test yapılırken cihazda Android 8.1 işletim sistemi yüklüdür. Tezin bundan sonraki bölümlerinde "1 numaralı gerçek test cihazı" olarak isimlendirilecektir ("Xiaomi Mi 8 Lite En Ucuz Fiyat ve Özellikleri - Epey," 2022).

Google Pixel 3a isimli telefondan da piyasada biri 5.6 İnç diğeri 6 İnç olmak üzere 2 farklı versiyon bulunmaktadır. Bu testte kullanılan versiyon 5.6 İnçlik versiyondur. Cihazda 5.6 İnç, OLED teknolojili, 1080x2220 (FHD+) Piksel çözünürlüğe ve 60 Hz ekran yenileme hızına sahip bir ekran bulunmaktadır. 2.0 GHz frekansa sahip 8 çekirdekli Qualcomm Snapdragon 670 (SDM670) işlemciye sahip olan bu cihazda 4 GB'lık bir ön bellek bulunmaktadır. Test yapılırken cihazda yüklü olan işletim sistemi Android 9'dir. Tezin bundan sonraki bölümlerinde "2 numaralı gerçek test cihazı" olarak isimlendirilecektir ("Google Pixel 3a En Ucuz Fiyat ve Özellikleri - Epey," 2022).

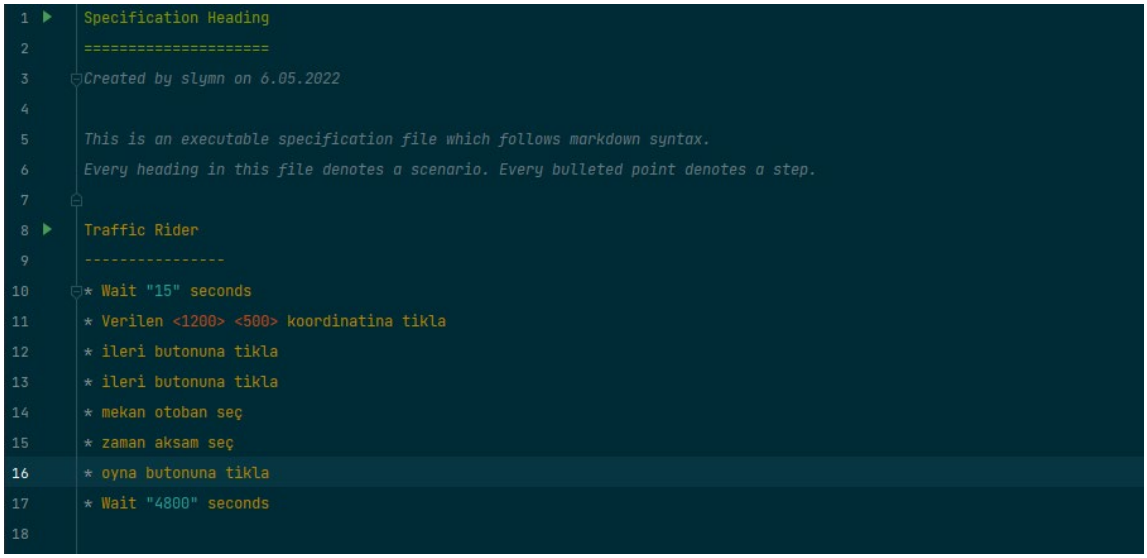
Samsung S10 isimli telefondan piyasada tek versiyon bulunmaktadır. Test cihazımızda, 6.4 İnç, dinamik AMOLED teknolojili, 2280x1080 (FHD+) Piksel çözünürlüğe ve 60 Hz ekran yenileme hızına sahip bir ekran bulunmaktadır. 2.73 GHz frekansa sahip Samsung Exynos 9 Series 9820 ana işlemci 2 adet 2.31 GHz frekansa sahip ARM Cortex-A75 yarımcı işlemci ve 4 adet de 1.95 GHz frekansa sahip ARM Cortex-A55 2. yardımcı işlemci olmak üzere toplam 8 çekirdekli işlemciye sahip olan bu cihazda 6 GB'lık bir ön bellek bulunmaktadır. Test yapılırken cihazda yüklü olan işletim sistemi Android 10'dur. Tezin bundan sonraki bölümlerinde "3 numaralı gerçek test cihazı" olarak isimlendirilecektir ("Samsung Galaxy S10 (SM-G973F) En Ucuz Fiyat ve Özellikleri - Epey," 2022).

Bütün cihazlarda oyunun bütün ilerlemelerini açmak için hileli bir versiyonu yüklenmiş ve oyun o şekilde test edilmiştir.

Normalde günlük kullanım için kullanılan telefonlarda arka planda sürekli işlem yapan uygulamalar kullanılmaktadır. Test uygulanırken telefonun gündelik hayatta kullanımını simüle edebilmek için cihazın ekran kartına ve işlemcisine sürekli işlemler yaptıracak Heater SQZSoft isimli uygulama test boyunca bütün cihazlarda yüksek derecede açık şekilde testler gerçekleştirilmiştir.

4.2. Gerçek Cihazlarda Test

Test cihazları otomasyon testlerin yapılacağı bilgisayara bağlantıları gerçekleştirilerek kilit ekranı açık bir şekilde teste hazır hale getirilmiştir. Testten önce cihazlarımızın şarj seviyesi %100'dür. Appium Server GUI ve Appium Inspector uygulamaları aracılığıyla telefonlara bağlanılmıştır. Önceden belirlenen test senaryolarına uygun şekilde yazılan Java tabanlı otomasyon test yazılımına ait görüntü Şekil 10.'de gösterilmektedir.



Şekil 10. Test Yazılımına Ait Örnek Kod

4.2.1. 1 Numaralı Gerçek Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp, Tablo 1. ve Tablo 2.'de alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar akıcı olarak geçmediği gözlenmiştir. Test cihazı, bazı test senaryolarını çalıştırırken 80 dakikalık test süresinin sonuna ulaşamadan uygulama hata vermiş ve çökmüştür.

Tablo 1.

1 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (Dakika)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	29,43	28,81	30,10	31,53	29,81	27,38
10	63,87	61,29	60,43	60,28	62,02	62,11
15	94,4	93,12	92,73	91,7	93,35	94,96
20	127,91	123,16	123	127,37	123,65	123,52
25	164,94	161,19	162,08	160,98	162,65	163,3
30	181,62	183,5	180,67	181,57	184,08	182,85
35	214,73	212,18	214,03	216,12	214,27	214,48
40	240,83	240,41	238,58	239,24	242,9	239,24
45	267,15	267,35	266,86	268,24	266,58	266,45
50	294,58	296,63	296,42	293,1	293,92	295,44
55	325,36	325,55	325,48	328,15	326,33	327,27
60	359,94	357,49	356,49	357,68	359,1	360,91
65	393,82	394,52	392,63	393,97	393,99	395,95
70	*	*	*	*	*	*
75	*	*	*	*	*	*
80	*	*	*	*	*	*

*= uygulama çöktüğü için veri alınamamıştır.

Tablo 2.

1 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	29,38	28,99	28,54	28,25	27,84	27,17
10	63,12	63,72	63,89	60,37	59,15	60,72
15	95,75	94,21	94,99	93,26	94,64	94,55
20	125,74	125,65	124,68	125,12	123,40	124,43
25	163,5	162,6	161,39	163,19	163,63	164,49
30	181,81	183,67	184,88	180,38	180,98	182,78
35	213,06	216,72	216,44	216,93	214,69	212,10
40	240,73	238,85	239,66	240,14	242,68	239,23
45	269,92	268,15	266,19	268,11	268,26	269,32
50	293,41	292,64	293,05	294,17	296,18	295,55
55	328,23	327,71	328,93	327,5	327,95	325,63
60	356,82	359,41	360,25	358,34	359,01	357,64
65	394,63	393,32	395,04	394,11	393,82	392,91
70	*	*	*	*	*	*
75	*	*	*	*	*	*
80	*	*	*	*	*	*

*= uygulama çöktüğü için veri alınamamıştır.

4.2.2. 2 Numaralı Gerçek Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp, Tablo 3. ve Tablo 4.'de alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar akıcı olarak geçmediği gözlenmiştir. Test cihazı,

bazı test senaryolarını çalıştırırken 80 dakikalık test süresinin sonuna ulaşamadan uygulama hata vermiş ve çökmüştür.

Tablo 3.

2 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (Dakika)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	39,53	34,46	34,63	36,57	39,83	38,20
10	67,23	66,76	66,42	70,92	69,78	69,33
15	105,02	104,91	105,93	106,16	102,82	104,70
20	137,06	138,95	138,53	138,91	139,26	141,55
25	178,54	179,77	175,41	176,01	177,14	176,07
30	206,53	206,21	206,19	205,82	207,94	205,91
35	237,31	239,92	239,18	237,16	237,16	240,41
40	269,07	271,16	270,54	270,44	268,39	271,42
45	297,94	296,33	300,26	297,38	301,79	300,57
50	327,42	327,69	330,65	328,72	332,23	333,43
55	367,89	366,15	368,45	367,96	366,22	366,31
60	397,13	397,63	397,15	397,87	397,97	397,80
65	431,21	432,12	434,79	431,46	441,80	442,21
70	*	*	*	476,14	475,06	472,13
75	*	*	*	501,2	501,4	504,81
80	*	*	*	549,73	545,07	549,34

*= uygulama çöktüğü için veri alınamamıştır.

Tablo 4.

2 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	38,03	33,50	34,13	34,81	33,05	34,50
10	66,44	66,54	65,22	66,88	67,12	66,55
15	102,31	102,47	104,68	100,89	103,45	100,41
20	140,72	138,51	135,56	138,30	138,47	137,58
25	179,25	176,1	175,02	177,75	175,34	179,11
30	206,91	201,04	201,86	204,77	204,01	201,73
35	239,96	235,93	236,01	237,19	237,56	238,39
40	271,20	266,49	266,19	269,25	268,28	268,70
45	299,62	298,5	299,31	297,91	298,05	298,08
50	330,32	327,74	327,47	328,29	328,12	328,62
55	366,43	363,51	364,88	360,08	362,39	362,78
60	396,05	395,73	397,51	396,13	396,61	399,09
65	435,25	435,05	436,76	435,32	434,98	437,12
70	*	*	*	*	*	*
75	*	*	*	*	*	*
80	*	*	*	*	*	*

*= uygulama çöktüğü için veri alınamamıştır.

4.2.3. 3 Numaralı Gerçek Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp Tablo 5. ve Tablo 6.'da alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar akıcı olarak geçmediği gözlenmiştir. Test cihazı,

bazı test senaryolarını çalıştırırken 80 dakikalık test süresinin sonuna ulaşamadan uygulama hata vermiş ve çökmüştür.

Tablo 5.

3 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (Dakika)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	35,54	33,62	37,29	35,19	33,08	34,79
10	72,2	73,85	73,11	73,55	73,31	75,45
15	109,76	113,55	111,61	111,03	112,85	112,55
20	148,83	148,42	147,2	147,08	148,21	150,67
25	192,42	193,38	194,12	194,62	191,42	190,44
30	224,74	223,2	223,67	224,11	223,01	222,7
35	262,89	261,9	258,82	262,95	260,63	261,85
40	295,91	294,63	296,8	296,26	294,2	293,96
45	327,06	326,86	326,63	328,99	325,18	326,03
50	358,47	360,09	362,24	360,59	361,06	362,96
55	400,53	396,44	398,29	400,11	400,54	396,95
60	434,2	437,69	435,01	437,22	436,22	438,2
65	482,94	485,66	485,27	485,73	482,53	483,87
70	*	*	*	518,20	519,57	520,97
75	*	*	*	549,32	552,75	551,12
80	*	*	*	596,98	599,46	596,42

*= uygulama çöktüğü için veri alınamamıştır.

Tablo 6.

3 Numaralı Gerçek Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	33,06	33,68	34,26	33,73	37,81	37,81
10	72,63	74,56	72,18	75,65	73,14	72,09
15	109,44	109,75	113,68	112,61	110,95	111,89
20	150,62	147,53	149,87	151,53	150,34	148,08
25	194,60	191,49	193,79	194,2	194,76	194,48
30	220,76	223,78	221,11	223,2	222,30	220,49
35	260,37	262,85	258,22	261,11	259,83	260,37
40	294,55	295,62	292,93	295,26	293,82	294,35
45	325,91	328,69	327,76	329,29	328,99	325,18
50	358,11	360,48	359,62	359,76	361,97	362,33
55	397,1	396,5	399,62	398	400,64	397,59
60	436,65	434,25	435,59	438,31	435,27	436,61
65	484,38	486,08	483,6	482,04	486,52	484,74
70	*	*	*	*	*	*
75	*	*	*	*	*	*
80	*	*	*	*	*	*

*= uygulama çöktüğü için veri alınamamıştır.

Test yapılan cihazlar farklı ön bellek ve işlemcilere sahip olduğu için beklenildiği gibi alınan yol ortalaması her telefonda farklı çıkmıştır. Test senaryosu 4, 5 ve 6'daki sonuçlara göre çöl yolunda telefonlarda çökme yaşanmamıştır. Uygulamada çöl yolu ile diğer yollar arasındaki temel fark 3 boyutlu nesnelerin sayı, boyut ve sıklığıdır.

4.3. Sanal Cihazlarda Test

Tezin ana konusu olan yazılım testinde gerçek cihazlarla sanal cihazlar arasındaki farkın bulunabilmesi için Bölüm 4.2'de testleri gerçekleştirilmiş gerçek cihazların emülatör yardımıyla sanal bir simülasyonu oluşturulmuştur. Bu simülasyonlar için Android Studio ve Genymotion uygulamaları kullanılmıştır. Uygulamaya gerçek cihazların bölüm 4.1'de yer alan bilgilere göre sanal cihaz bilgileri girilerek simülasyon test için çalıştırılmıştır. Emülatörlerde işlemci ayarlama gibi bölüm bulunmamaktadır bu yüzden test için kullanılacak bilgisayarın işlemci hızı telefonların işlemci hızına düşürülmüştür. Test cihazlarında batarya seçenekleri; batarya durumu iyi, batarya doluluğu %100 ve şarja takılı olarak ayarlanmıştır.

4.3.1. 1 Numaralı Sanal Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp Tablo 7. ve Tablo 8.'de alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar gerçek cihaza göre daha akıcı olsa 1 numaralı sanal cihazda da kasmalar meydana geldiği gözlenmiştir. Test cihazı, gerçek test cihazının aksine 80 dakikalık test süresinin sonuna kadar test devam etmiş ve herhangi bir hata ile karşılaşılmamıştır.

Tablo 7.

1 Numaralı Sanal Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	48,27	47,21	48,11	49,87	46,59	53,34
10	97,96	95,73	99,69	95,09	100,97	94,85
15	143,44	145,59	149,84	145,37	151,05	143,28
20	193,94	192,68	193,54	193,19	192,07	190,22
25	237,62	240,79	242,8	239,44	241,09	241,87
30	285,22	289,19	285,61	288,83	291,65	294,56
35	332,46	335,34	332,5	334,48	335,24	338,9
40	382,68	383,04	380,23	381,08	381,07	383,75
45	429,44	428,88	431,41	432,66	434,2	428,43
50	475,83	479,6	479,66	474,68	474,15	475,11
55	527,28	526,81	524,6	525,07	523,51	527,96
60	571,01	574,53	573,57	572,78	575,63	571,57
65	622,55	625,45	627,43	619,96	624,28	618,22
70	673,08	673,29	665,9	674,98	674,29	666,45
75	713,87	715,58	715,3	719,59	713,85	714,54
80	759,37	774,51	765,24	773,86	764,69	772,11

Tablo 8.

1 Numaralı Sanal Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	54,09	50,28	46,36	48,96	53,02	52,86
10	98,62	98,67	102,64	96,17	96,05	102,75
15	142,47	147,99	144,53	149,56	147,99	144,43
20	191,92	194,35	197,45	194,79	197,63	196,67
25	241,36	240,87	237,72	240,58	237,23	240,78
30	286,18	286,72	286,9	288,34	292,97	292,71
35	334,8	335,21	335,4	340,91	339,64	339,89
40	384,78	383,21	383,84	380,31	387,94	386,21
45	432,36	433,16	429,87	432,66	434,6	428,47
50	479,97	473,38	478,35	476,01	473,61	477,34
55	524,48	524,99	523,83	523,88	528,74	526,29
60	578,98	574,64	573,25	573,97	574,37	577,74
65	623,52	627,11	623,16	623,46	620,21	620,93
70	673,14	668,74	674,05	667,57	671,81	673,56
75	722,62	720,69	719,82	719,25	719,06	713,43
80	769,39	762,97	759,79	768,38	759,27	773,51

4.3.2. 2 Numaralı Sanal Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp, Tablo 9. ve Tablo 10.'da alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar gerçek cihaza göre daha akıcı olsa 2 numaralı sanal cihazda da kasmalar meydana geldiği gözlenmiştir. Test cihazı, gerçek test cihazının

aksine 80 dakikalık test süresinin sonuna kadar test devam etmiş ve herhangi bir hata ile karşılaşılmamıştır.

Tablo 9.

2 Numaralı Sanal Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	47,70	48,82	46,85	46,33	47,53	51,56
10	95,37	101,21	98,15	95,31	101,23	97,24
15	143,18	148,73	148,81	147,15	143,61	148,65
20	190,98	190,02	194,99	190,86	189,85	193,65
25	238,76	245,35	242,15	243,94	238,62	237,19
30	284,32	291,15	290,87	287,36	290,22	293,52
35	330,10	332,47	333,42	337,01	335,51	338,89
40	374,79	381,01	383,95	384,08	387,97	385,51
45	420,64	430,52	434,75	434,01	430,55	430,91
50	468,32	476,67	476,12	479,15	478,48	473,27
55	515,71	527,26	529,69	525,84	529,20	528,91
60	563,52	576,73	578,43	578,29	572,72	574,08
65	611,35	620,79	627,32	621,21	622,82	626,16
70	656,51	669,56	666,08	668,71	667,86	665,02
75	722,58	721,47	717,54	719,05	714,36	714,23
80	763,66	773,02	760,26	774,41	764,66	770,38

Tablo 10.

2 Numaralı Sanal Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	48,21	53,25	48,09	47,79	53,43	46,93
10	101,27	94,49	95,41	99,87	102,94	100,58
15	146,20	149,08	150,21	148,32	151,79	143,06
20	197,34	189,44	193,11	193,66	194,03	195,99
25	238,73	244,79	241,45	242,21	245,09	241,34
30	290,29	288,08	294,16	286,69	286,65	292,47
35	341,01	332,06	340,09	334,15	334,34	333,65
40	386,79	381,44	388,49	388,85	381,19	382,19
45	432,37	429,53	434,28	430,89	431,17	433,28
50	475,64	478,80	479,05	478,32	478,56	478,77
55	525,50	529,27	529,89	523,53	528,49	529,67
60	574,59	571,39	577,87	576,06	575,69	571,02
65	625,41	625,31	619,61	621,83	624,09	620,58
70	673,85	667,15	665,23	673,69	667,14	673,74
75	722,67	721,06	719,51	721,54	714,46	716,43
80	759,79	763,99	767,23	770,13	774,99	769,39

4.3.3. 3 Numaralı Sanal Test Cihazı

Uygulama otomasyon ile teste başlamış, test senaryosuna göre istenen özelliklerdeki bölüme girmiş ve oyun başlamıştır. Oyunda sağ üst köşede bulunan katedilen kilometre göstergesi 80 dakikalık test süresince her 5 dakikada bir kayıt altına alınıp, Tablo 11. ve Tablo 12.'de alınan kayıtlar gösterilmektedir. Oyun içinde bulunan ağaç, ev, tünel, aydınlatma direkleri ve yol çizgileri gibi animasyonlar gerçek cihaza göre daha akıcı olsa 3 numaralı sanal cihazda da kasmalar meydana geldiği gözlenmiştir. Test cihazı, gerçek test

cihazının aksine 80 dakikalık test süresinin sonuna kadar test devam etmiş ve herhangi bir hata ile karşılaşılmamıştır.

Tablo 11.

3 Numaralı Sanal Test Cihazına Ait Test Sonuçları

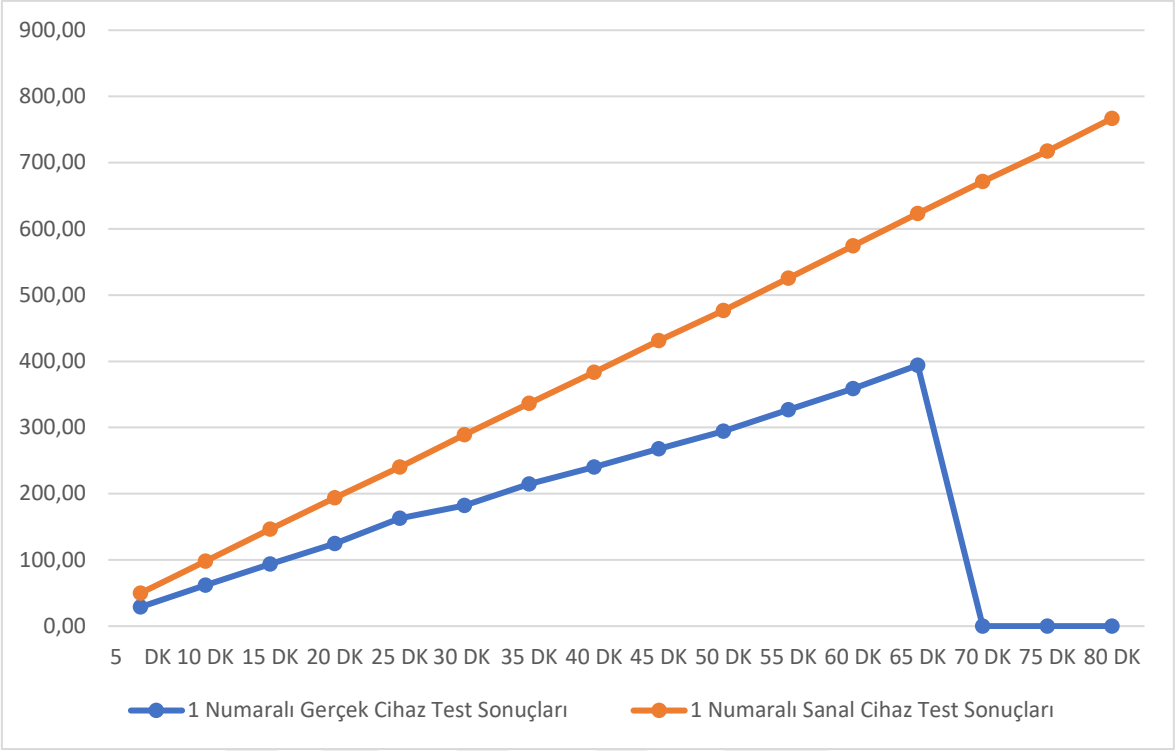
TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS01 (Kilometre)	TRTS02 (Kilometre)	TRTS03 (Kilometre)	TRTS04 (Kilometre)	TRTS05 (Kilometre)	TRTS06 (Kilometre)
5	56,59	55,99	57,84	57,52	55,92	57,63
10	102,47	101,53	101,45	99,52	102,54	99,58
15	154,53	151,93	152,37	151,72	157,16	154,49
20	200,83	200,46	197,13	198,95	197,53	202,56
25	242,81	244,04	240,08	246,81	243,59	246,28
30	298,15	298,03	295,44	292,84	291,91	293,35
35	340,87	346,39	345,42	345,81	339,94	348,19
40	393,05	387,03	389,66	389,63	387,34	394,27
45	440,76	441,51	438,64	439,52	441,22	436,69
50	487,91	485,71	486,76	488,89	484,83	487,99
55	538,31	536,04	538,06	535,33	539,07	537,85
60	580,59	578,15	577,04	584,46	579,04	577,15
65	629,88	634,99	630,42	635,41	634,36	635,51
70	680,14	677,02	678,23	682,22	675,28	679,59
75	719,72	714,89	713,14	714,01	720,67	714,76
80	758,23	766,13	772,51	771,83	764,39	772,72

Tablo 12.

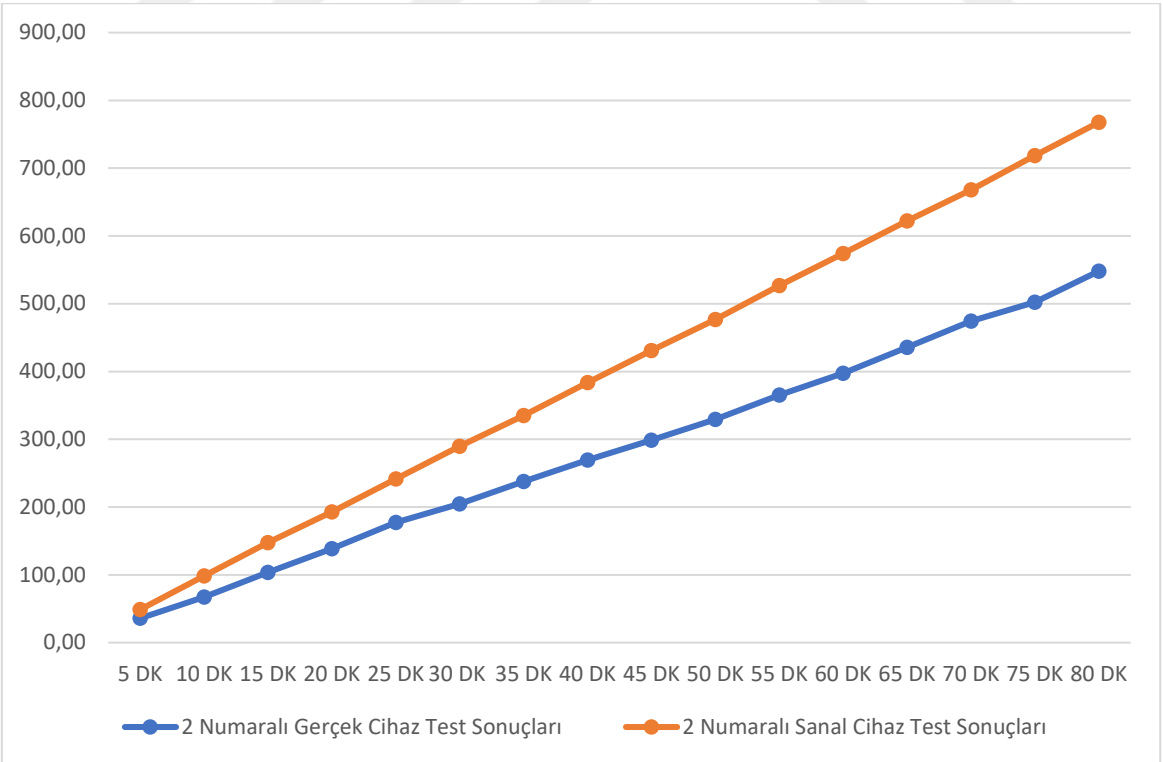
3 Numaralı Sanal Test Cihazına Ait Test Sonuçları

TEST SENARYOLARI / TEST SÜRESİ (DAKİKA)	TRTS07 (Kilometre)	TRTS08 (Kilometre)	TRTS09 (Kilometre)	TRTS10 (Kilometre)	TRTS11 (Kilometre)	TRTS12 (Kilometre)
5	52,95	58,13	56,17	50,77	59,34	51,13
10	106,31	104,75	108,18	105,39	108,41	99,91
15	156,18	156,94	149,68	157,11	156,36	156,83
20	198,65	202,69	205,55	197,78	200,64	205,76
25	240,37	244,89	245,12	242,66	242,61	248,63
30	292,97	292,38	292,97	293,09	294,73	291,30
35	341,57	341,28	347,11	340,65	343,46	339,41
40	385,82	393,87	393,96	394,75	391,79	392,97
45	433,5	438,02	435,92	439,3	439,86	436,25
50	485,17	481,45	480,31	485,32	487,99	482,72
55	538,16	538,43	534,96	533,82	536,91	534,45
60	578,41	586,32	582,11	582,47	580,21	581,84
65	631,72	631,48	634,06	634,89	634,46	629,92
70	680,74	677,33	680,13	675,31	682,81	676,02
75	719,34	721,35	716,45	718,45	716,76	715,12
80	755,59	757,69	757,38	773,77	762,81	774,41

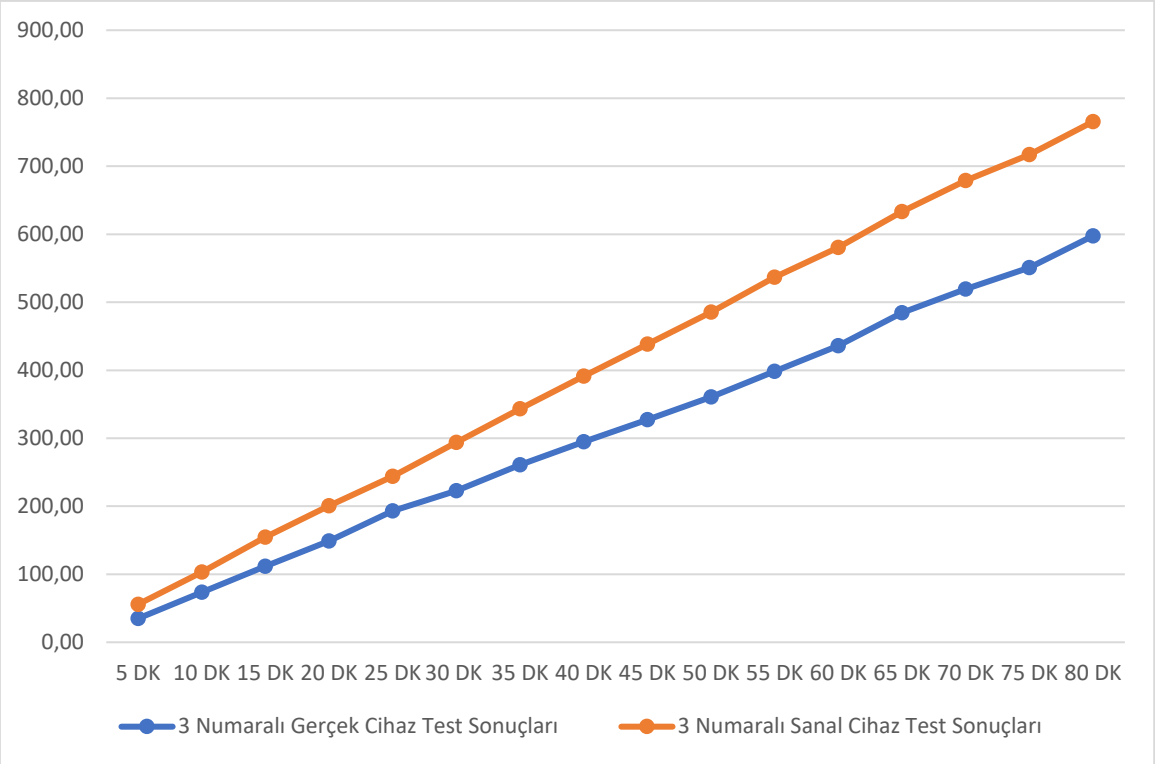
Sanal test cihazlarında gerçekleştirilen testlerde testin 80. dakikasında katedilen yol diğer sanal test cihazlarıyla karşılaştırıldığında yaklaşık değerler olduğu görülmektedir. Gerçek cihazlarda ortalama alınan yol cihazdan cihaza değişkenlik gösterirken, sanal cihazlarda fark olmaması ve sanal cihaz dediğimiz emülatorlerin gerçek cihazları tam olarak simüle edemediğini göstermektedir. Aynı özelliklerdeki gerçek ve sanal cihazların tüm test senaryolarına göre uygulanan test sonuçlarının ortalamaları Şekil 11, 12 ve 13'de verilmektedir.



Şekil 11. 1 Numaralı Gerçek ve Sanal Test Cihazlarının Test Sonuçlarının Ortalamalarının Karşılaştırılması



Şekil 12. 2 Numaralı Gerçek ve Sanal Test Cihazlarının Test Sonuçlarının Ortalamalarının Karşılaştırılması



Şekil 13. 3 Numaralı Gerçek ve Sanal Test Cihazlarının Test Sonuçlarının Ortalamalarının Karşılaştırılması

Şekillerde verilen ortalamalar alınırken gerçek cihazlarda çökme olan senaryolardaki dakikalar boş veri olarak alınmış ve var olan verilere göre ortalama alınmıştır. Grafikleri oluşturulmasındaki temel neden; sanal cihazların, gerçek cihazları ne kadar doğru bir şekilde simüle ettiğini ve test işlemlerinde alınacak çıktıların gerçek cihazlarda alınacak çıktılarla olası farklarını göstermektir.

Şekil 11. incelendiğinde 1 numaralı sanal ve gerçek cihaz test sonuçları arasındaki fark açıkça belli olmaktadır. 1 numaralı gerçek test cihazında tüm senaryolarda 65. dakikadan sonra çökme yaşanmış ve veri alınmamıştır. Ayrıca 1 numaralı gerçek test cihazına göre üretilmiş 1 numaralı sanal cihazda motorun aldığı yolun düzenli bir artış göstermesinden uygulamanın test süresince cihazda herhangi bir performans kaybı yaşanmadığı da görülmektedir. Ancak gerçek test cihazımızda motorun almış olduğu yol düzensiz artış göstermiş bazı 5 dakikalık dilimlerde ortalama 36 kilometre yol almışken bazı 5 dakikalık dilimde ise ortalama 22 kilometrelere kadar düşmüştür. Bu sonuç gerçek test cihazımızın testlerin gerçekleştiği sırada performans kaybı yaşadığını göstermektedir.

2 numaralı gerçek test cihazında çöl yolu seçiminde test süresince çökme yaşanmamıştır. Geri kalan senaryolar çöktüğü için Şekil 12.'de 65. dakikadan sonra gösterilen değerler çöl yolu seçimine ait üç test senaryosunun ortalamasına aittir. Diğer 9 test senaryosuna ait sonuçların 70, 75 ve 80'inci dakikalardaki verileri gösterilememiştir. Ayrıca 2 numaralı gerçek test cihazına göre üretilmiş 2 numaralı sanal cihazda motorun aldığı yolun düzenli bir artış göstermesinden uygulamanın test süresince cihazda herhangi bir performans kaybı yaşanmadığı da görülmektedir. Ancak gerçek test cihazımızda motorun almış olduğu yol düzensiz artış göstermektedir. Bu da gerçek test cihazımızın testlerin gerçekleştiği sırada performans kaybı yaşadığını göstermektedir.

3 numaralı gerçek test cihazında çöl yol seçiminde tüm zamanlarda çökme yaşanmamıştır. Geri kalan senaryolar çöktüğü için Şekil 13.'de 65. dakikadan sonra gösterilen değerler çöl yolu seçimine ait üç test senaryosunun ortalamasına aittir. Diğer 9 test senaryosuna ait sonuçların 70, 75 ve 80'inci dakikalardaki verileri gösterilememiştir. Ayrıca 3 numaralı gerçek test cihazına göre üretilmiş 3 numaralı sanal cihazda motorun aldığı yolun düzenli bir artış göstermesinden uygulamanın test süresince cihazda herhangi bir performans kaybı yaşanmadığı da görülmektedir. Ancak gerçek test cihazımızda motorun almış olduğu yol düzensiz artış göstermektedir. Bu da gerçek test cihazımızın testlerin gerçekleştiği sırada performans kaybı yaşadığını göstermektedir.

Bütün şekiller birlikte incelendiğinde 3 numaralı gerçek cihazın diğer gerçek cihaza göre sanal cihazlara daha yakın bir performans göstermekteyiz. Bu sonuç uygulamanın cihaz donanımlarına da bağlı olduğunu göstermektedir.

Göründüğü üzere sanal cihazlar, gerçek cihazları simüle ederken bire bir simüle edemezler. Bunun sebebi cihazların içerisindeki donanım farklılıklarıdır. Simülasyonlarda ön bellek boyutları, hafıza boyutları, ekran çözünürlük ve büyüklükleri gibi donanımsal konum, arama, sms, bildirimler vb. yazılımsal özellikleri simüle etme özellikleri yer alsa da bilgisayarda bulunan donanımlarla tam anlamıyla mobil cihazların donanımları simüle edilemediği görülmektedir. Bu sorunun en temel sebebi bilgisayar donanımları daha büyüktür, bu da soğutma için yeterli alana sahip oldukları anlamına gelir. Ayrıca bilgisayarlarda özel soğutma sistemleri vardır.

İşlemci, ram, ekran kartı gibi elektronik parçalar; silisyum, germanyum veya uygun diğer yarıiletken karışımlar kullanılarak üretilen transistörlerden üretilir. Çiğdemoğlu'na (1983) göre transistörler içerisinde bulunan silisyum gibi yarı iletken maddelerin en verimli çalışma sıcaklığı 25 santigrat derecedir ve sıcaklık ideal derecenin üzerine çıktıkça performans düşmektedir. Telefon veya bilgisayardaki işlemciler çalıştıkça ısınır ve 25 santigrat derecenin üzerinde her derece artışı performans azalmasına yol açar. Bilgisayarlarda bulunan fanlı soğutma sistemleri gibi donanımlar mobil cihazlara konulamadığı için mobil cihazlar yük altında bilgisayarlar gibi hızlı soğuyamazlar. Bu da performansa etki eden önemli bir faktördür.

BEŞİNCİ BÖLÜM
SONUÇ VE ÖNERİLER

Düşük bütçe ile üretilmeye çalışılan mobil uygulamalar, test işlemleri için emülatörler yardımı ile oluşturulan sanal cihazlarda test işlemlerini gerçekleştirebilirler. Sanal cihazlarda test; görünüş, ekran yerleşimi, renk, işleyiş vb. işlemleri test etmek için kullanılabilmektedir. Ancak performans odaklı testlerde sanal cihaz kullanımı geliştirici ve test mühendislerini yanıltacağı görülmektedir. Uygulamaların pazarda son kullanıcıya hitap ederken sorun çıkarmaması ve olumlu bildirimler alınabilmesi için gerçek cihazlarda test edilmesi gerekir. Gerçek cihazlarda test maddi açıdan uygulama bütçesini artırsa da sonradan çıkabilecek büyük sorunların önüne geçmek için önemlidir. Uygulama piyasaya sürüldükten sonra çıkan büyük hatalar yayıncı ve geliştirici için daha büyük maliyetler ortaya çıkarabilir.

Bu çalışmada, seçilen 1 mobil uygulama hem sanal cihazlarda hem de gerçek cihazlarda aynı otomasyon yazılımı ile test edilmiştir. Uygulamanın verdiği tepkiler ve testin gerçekleştirilme süreleri karşılaştırılarak uygun test yöntemi gösterilmeye çalışılmıştır. Çıkan sonuçlara bakarak mobil otomasyon testlerinin, maliyetlerine rağmen gerçek cihazlar üzerinde yapılmasının doğru sonuçlar alabilmek adına daha verimli olduğu kanıtlanmıştır.

Amacı mobil uygulamaların otomasyon kullanılarak gerçek ve sanal cihazlarda test edilmesi ve karşılaştırılması olan çalışmamız, otomasyonla mobil uygulama test etmek isteyen test mühendislerine hem literatür anlamında hem de test cihazı seçimi için tam bir rehber olacaktır. Ayrıca çalışma mobil uygulamalarda test üzerine faaliyet gösteren firmalar için de önemli sonuçlar ortaya koymaktadır. Test işlemlerinde sanal cihaz kullanımının uygulama ve geliştirici firma için ne gibi sonuçlar doğurabileceği testlerle gösterilmektedir.

Bu çalışma sadece Android işletim sistemine sahip cihazları, Microsoft işletim sistemine sahip cihazla test etmiştir. Bu sonuçlar diğer işletim sistemleri için aynı etkiyi vermeyebilir. Bu çalışmada yapılan testi, iOS işletim sistemini kullanan ve MacOS işletim sistemi ile çalışan cihazlarda da test etmek gerekir. Bu çalışmanın devamında, Apple marka cihazların birbirine olan uyumluluğu ve birbirleri arasında kurulan ekosistemin göstereceği pozitif veya negatif etkilerin çalışılması planlanmaktadır.

KAYNAKÇA

Ballard, B. (2007). *Designing the Mobile User Experience*. Wiley. https://www.wiley.com/en-uz/9780470060582 adresinden erişildi.

Berner, S., Weber, R. ve Keller, R. K. (2005). Observations and lessons learned from automated testing. *Proceedings - International Conference on Software Engineering* içinde (C. 2005, ss. 571-579). doi:10.1109/ICSE.2005.1553603

Chan, H. A. (2004). Accelerated stress testing for both hardware and software. *Proceedings of the Annual Reliability and Maintainability Symposium*, 346-351. doi:10.1109/RAMS.2004.1285473

Chauhan, N. S. ve Saxena, A. (2013). A green software development life cycle for cloud computing. *IT Professional*, *15*(1), 28-34. doi:10.1109/MITP.2013.6

Çiğdemoğlu, M. (t.y.). Yüksek güçlü elektronik devre elemanları için soğutucu tasarımı. *Emo*.

Dustin, E., Rashka, J. ve Paul, J. (1999). *Automated Software Testing: Introduction, Management, and Performance*. *Addison-Wesley Professional*. Addison-Wesley Professional.

Fewster, M. ve Graham, D. (1999). *Software test automation : effective use of test execution tools*. Addison-Wesley.

Gao, J., Bai, X., Tsai, W. T. ve Uehara, T. (2014). Mobile application testing: A tutorial. *Computer*, *47*(2), 46-55. doi:10.1109/MC.2013.445

Genymotion – Android Emulator for app testing Cross-platform Android Emulator for manual and automated app testing. (t.y.). 5 Ağustos 2022 tarihinde https://www.genymotion.com/ adresinden erişildi.

Gökgöz, B. ve Keskinkılıç, M. (2018). Yazılım Proje Geliştirme Sürecinde Proje Yönetim Aşamaları ve Risk Analizi İncelemesi. *SETSCI Conference Indexing System*, *3*, 1040-1045.

Hamilton, T. (2022a, 16 Nisan). STLC (Software Testing Life Cycle) Phases, Entry, Exit Criteria. *Guru99*. 21 Mayıs 2022 tarihinde https://www.guru99.com/software-testing-life-cycle.html adresinden erişildi.

Hamilton, T. (2022b, 23 Nisan). What is STRESS Testing in Software Testing? Tools, Types, Examples. 26 Haziran 2022 tarihinde https://www.guru99.com/stress-testing-tutorial.html adresinden erişildi.

Hancı, A. K. (2017). *Yazılım Testi Tasarım Tekniklerinin Matematiksel Model İle Analizi*. İstanbul Üniversitesi.

Kaner, Cem., Falk, J. L. ve Nguyen, H. Quoc. (1993). *Testing Computer Software* (C. 1). Van Nostrand Reinhold. https://books.google.com/books/about/Testing_Computer_Software.html?hl=tr&id=67JQAAAAMAAJ adresinden erişildi.

Kaymak, A. (2020). *Comparison of Software Testing Tools and Selection of Automation Over Manual Testing*. http://acikerisim.ybu.edu.tr:8080/xmlui/handle/123456789/2633 adresinden erişildi.

Koomen, Tim. ve Pol, Martin. (1999). *Test process improvement: a practical step-by-step guide to structured testing*. Addison-Wesley.

Koziokas, P. T., Tselikas, N. D. ve Tselikis, G. S. (2017). Usability testing of mobile applications: Web vs. Hybrid Apps. *ACM International Conference Proceeding Series*, *Part F132523*. doi:10.1145/3139367.3139410

Miller, R. W. ve Collins, C. T. (2001). Acceptance Testing. *Proc. XPUniverse*.

Nalbant, E. (2020). *Yazılım yaşam döngüsünde testin önemi ve bir test otomasyonunun gerçekleştirilmesi*. *Yıldız Teknik Üniversitesi / Fen Bilimleri Enstitüsü / Elektronik ve Haberleşme Mühendisliği Ana Bilim Dalı / Haberleşme Bilim Dalı*. https://acikbilim.yok.gov.tr/handle/20.500.12812/378761 adresinden erişildi.

Rice, R. W. (2003). Surviving the Top Ten Challenges of Software Test Automation. Rice Consulting Solutions, LLC. www.riceconsulting.com/cheaptools.htm. adresinden erişildi.

Run apps on the Android Emulator. (t.y.). 22 Mayıs 2022 tarihinde https://developer.android.com/studio/run/emulator?gclid=Cj0KCQjwvqeUBhCBARIsAOdt45YoHAdaBQgRSlXGF9sod1wrBoqWtPBFMmt6mFDvxpeggmFIpITRnNkaAqfUEALw_wcB&gclsrc=aw.ds adresinden erişildi.

Sarojadevi, H. (2011). Performance Testing Methodologies and Tools. *Journal of Information Engineering and Applications*, *1*(5). www.iiste.org adresinden erişildi.

Sertifikalı Test Uzmanı Temel Seviye Ders Programı. (2018). www.turkishtestingboard.org adresinden erişildi.

Set up for Android Development. (t.y.). 24 Mayıs 2022 tarihinde https://source.android.com/setup/intro?hl=en adresinden erişildi.

Smartphone Shipments Declined in the Fourth Quarter But 2021 Was Still a Growth Year with a 5.7% Increase in Shipments, According to IDC. (t.y.). 19 Mayıs 2022 tarihinde https://www.idc.com/getdoc.jsp?containerId=prUS48830822 adresinden erişildi.

K. Sneha and G. M. Malle, "Research on software testing techniques and software automation testing tools," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017, pp. 77-81, doi: 10.1109/ICECDS.2017.8389562.

Store Intelligence Data Digest. (2022).*Sensor Tower*. go.sensortower.com adresinden erişildi.

Şeker, S. E. (2015). Yazılım Geliştirme Modelleri ve Sistem/Yazılım Yaşam Döngüsü. *YBS Ansiklopedi*, *2*(3), 18-29.

Takgil, B. (2016). Android Mobil Uygulamalar İçin Yazılım Testi. *El-Cezerî Fen ve Mühendislik Dergisi*, *3*(2), 324-328. doi:10.31202/ECJSE.264196

Türk Dil Kurumu Sözlükleri. (t.y.). 5 Ağustos 2022 tarihinde https://sozluk.gov.tr/ adresinden erişildi.

Türkiye Yazılım Kalite Raporu (TSQR) - Turkish Testing Board. (t.y.). 5 Ağustos 2022 tarihinde https://www.turkishtestingboard.org/turkiye-yazilim-kalite-raporu/ adresinden erişildi.

Watkins, J. ve Mills, S. (2010). *Testing it: An Off-the-shelf software testing process, second edition*. *Testing It: An Off-the-Shelf Software Testing Process, Second edition* (C. 9780521148016). Cambridge University Press. doi:10.1017/CBO9780511997310

What is Software Testing and How Does it Work? | IBM. (t.y.). 5 Ağustos 2022 tarihinde https://www.ibm.com/topics/software-testing adresinden erişildi.

Yener, A.H., Baştürk, F. ve Meçik, B. (2019). Yazılım Geliştirme ve Test Otomasyon ile Verimlilik Artışı: General Mobile Productivity Increase with Software Development and Test Automation: General Mobile. KMÜ Mühendislik ve Doğa Bilimleri Dergisi, 1(1), 172-196.

Yoon, I. C., Sussman, A., Memon, A. ve Porter, A. (2008). Effective and scalable software compatibility testing. *ISSTA'08: Proceedings of the 2008 International Symposium on Software Testing and Analysis 2008*, 63-73. doi:10.1145/1390630.1390640

Zhan, W. ve Yan, G. (2019). Testing of mobile applications. A review of industry practices. www.bth.se adresinden erişildi.