T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

UZAKTAN EĞİTİMDE ÖLÇME DEĞERLENDİRME SİSTEMİ TASARIMI VE YAZILIM TEST TEKNİKLERİ İLE PERFORMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Bilg.Müh. Muhammed Maruf ÖZTÜRK

Enstitü Anabilim Dalı	**:**	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	**:**	**BİLGİSAYAR MÜHENDİSLİĞİ**
Tez Danışmanı	**:**	**Doç. Dr. Ahmet ZENGİN**

Haziran 2012

T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ

UZAKTAN EĞİTİMDE ÖLÇME DEĞERLENDİRME SİSTEMİ TASARIMI VE YAZILIM TEST TEKNİKLERİ İLE PEROFRMANS ANALİZİ

YÜKSEK LİSANS TEZİ

Bilg.Müh. Muhammed Maruf ÖZTÜRK

Enstitü Anabilim Dalı	:	**BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ**
Enstitü Bilim Dalı	:	**BİLGİSAYAR MÜHENDİSLİĞİ**

Bu tez 29 / 06 /2012 tarihinde aşağıdaki jüri tarafından Oybirliği ile kabul edilmiştir.

Prof. Dr. Hüseyin EKİZ	Prof. Dr. Nejat YUMUŞAK	Doç Dr. Ahmet ZENGİN
Jüri Başkanı	**Üye**	**Üye**

ÖNSÖZ

Bu tez yazılım geliştirme sürecinin farklı aşamalarında kullanılabilen ve kullanılması önerilen birçok test tekniğini açıklamakta ve bir ölçme değerlendirme sistemi üzerinde kullanımını örneklemektedir. Test süreci tümleşik süreç ve atik yazılım geliştirme süreci gibi yaygınlıkla kullanılan herhangi bir süreç olabilir. Ancak çalışmamızda V-modeli temel alınmıştır.

Geleneksel yazılım test etme teknikleri açıklanmış ve sistem seviyesinde uygulanabilecek performans testi, yükleme testi, birim testi ve kara kutu test teknikleri üzerinde durulmuştur.

Günümüzde kullanılan uzaktan eğitim sistemleri incelenmiş ve bu sistemlerde kullanılabilecek uygun bir ölçme değerlendirme sistemi geliştirilmiştir. Geliştirilen ölçme değerlendirme sisteminin tüm adımlarında uygulanması gereken yazılım test teknikleri açıklanmış, uygulanmış ve sonuçta hataları giderilmiş bir ölçme değerlendirme sistemi elde edilmiştir.

Gerçekleştirilen çalışmanın test tekniklerinin yazılım projelerinde kullanımını yaygınlaştırması ve ileride yapılacak uzaktan eğitim sistemi uygulamalarının daha güvenli ve tutarlı olması için bir temel teşkil etmesi amaçlanmıştır.

Bu tezin hazırlanmasında emeği geçen sayın Doç. Dr. Ahmet ZENGİN hocama katkılarından dolayı teşekkürlerimi sunarım. Çalışmam sırasında bana yardımcı olan ve desteğini hiçbir zaman eksik etmeyen aileme teşekkür ederim.

İÇİNDEKİLER

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

SİMGELER VE KISALTMALAR LİSTESİ

ABD	: Amerika Birleşik Devletleri
ASP	: Active server pages (etkin sunucu sayfaları)
ID	: Identity (Kimlik)
IDE	: Integrated Development Environemnt (Bütünleşik geliştirme ortamı)
IEEE	: Institute of Electrical and Electronics Engineers (Elektrik Elektronik Mühendisleri Enstitüsü)
HTML	: HyperText Markup Language
ÖDS	: Ölçme değerlendirme sistemi
SQL	: Structed Query Language
WWW	: world wide web

ÖZET

Anahtar kelimeler: Yazılım Testi, Test Teknikleri, V-model, Uzaktan Eğitim, Ölçme ve Değerlendirme Sistemi

Yazılım testi, bir yazılım sisteminin gereksinim ve beklenen hedefleri karşılayabilmesi için her ara üründe sistem üzerinde yapılan kontrol işlemleridir. Yazılım testinin amacı sistem hatalarını bulmak, çıkabilecek hataları önlemek ve sistemin beklenen fonksiyonları gerçekleştirmesini sağlamaktır. Yapılan çalışmada statik test, performans test ve sistem testi konularına ağırlık verilmiştir.

Bu çalışmada bankacılık, havacılık, uzay bilimleri ve savunma sanayi gibi alanlarda sıkça kullanılan yazılım test teknikleri uzaktan eğitim sistemi üzerinde uygulanmıştır. Yazılım testi tekniklerinden performans testi, birim testi, kara kutu testi ve beyaz kutu testi, gerileme testi ve stres testi konuları tasarlanan uzaktan eğitim değerlendirme sistemi üzerinde uygulanmıştır.

Yapılan testlerin, değerlendirme ve ölçme işlemlerinin sonuçları grafikler halinde sunulmuştur. Tüm testlerin sonuçlarına göre hatalar giderildikten sonra yazılım testinin son aşaması olan yükleme testi yapılmıştır.

Ölçme ve değerlendirme sistemi Asp.net 4.0 ve SQL SERVER 2005 araçları kullanılarak gerçekleştirilmiştir. Elde edilen sonuçlara göre daha tutarlı, yazılım adımları izlenebilen, sağlam ve kolay güncellenebilen bir uzaktan eğitim ölçme ve değerlendirme sistemi önerilmiştir. Yapılan çalışmanın uzaktan eğitimde yapılması düşünülen yazılım test çalışmalarına model olması hedeflenmektedir. Mobil uygulamalar günden güne yaygınlaştığı için gerçekleştirilen çalışma web tabanlı mobil cihazlara da uygulanabilir.

DISTANCE EDUCATION MEASUREMENT AND EVALUATION SYSTEM DESIGN AND PERFORMANCE ANALYSE WITH SOFTWARE TESTING TECHNIQUES

SUMMMARY

Keywords: Software Testing, Test Techniques, V-model, Distance Education Measurement and Evaluation System

Software testing is control process performed on every intermediate product in a system development to guarantee that a software system can meet the needs and goals. The purpose of software testing is to find system errors, avoid any mistakes and ensure that can perform expected functions. In this study, static, performance and system tests are considered.

In this work, various test techniques have been implemented on distance education system which are frequently used by banking, aviation, space science, defence industries. Software testing techniques such as performance testing, unit testing, black-box testing, regression testing, stress testing and white-box testing have been used for development of distance education measurement and evaluation system.

The results of the tests performed on evaluation and measurement system have been reported with graphs. After all errors and failures are removed from the developed system, load test has been performed.

Measurement and evaluation system is being realized using Asp.net 4.0 and Sql Server 2005 software tools. According to the results obtained from all performed tests, a more consistent, robust, manageable traceable measurement and evaluation system for distance education have been proposed. The study is aimed to be a model for distance education system testing. Since mobile applications are becoming widespread use, performed study can easily be applied to mobile devices.

BÖLÜM 1. GİRİŞ

1.1. Problem Tanımı

Tüm dünyada ve ülkemizde internet kullanım oranında ciddi bir artış gözlemlenmektedir. İnternet üzerinden yapılan işlemlerin yaygınlaşmasıyla birlikte çeşitli kurumlar sistemlerini internet ortamına taşımaya başlamıştır. Eğitim kurumları da web tabanlı eğitim sistemine geçiş yapmaya başlamıştır. Web tabanlı eğitimin mekandan bağımsız olması ve gelişen video ortam teknolojileri sayesinde örgün eğitimdeki sınıf ortamının birebir internet ortamına taşınması web tabanlı eğitimin tercih edilme nedenlerindendir [1].

Özellikle üniversitemizde çeşitli ön lisans, lisans ve lisansüstü eğitimler uzaktan eğitim ile verilmektedir. Çeşitli devlet kurumları da kendi personellerini uzaktan eğitim yoluyla hizmet içi eğitime almaya başlamışlardır.(Sağlık Bakanlığı USES, MEB Sertifika Temelli Uzaktan Hizmetiçi Eğitim Projesi (e-Sertifika), E.G.M Eğitim portalı, vb) . Bu eğitimlerin verilmesi için gerekli yazılımlar gerek ticari firmalar gerekse ünivesitelerde proje, tez vb. çalışmalar ile yapılmaktadır.

Geliştirilen uzaktan eğitim sistemleri web tabanlı yazılımlardır. Bu yazılımlarda video, grafik desteği ve docx, pdf gbi dosya formatlarında ders içerikleri bulunmaktadır. Halen kullanılmakta olan uzaktan eğitim sistemlerinde yazılım geliştirme, test ve analiz çalışmaları yeterince yapılmamaktadır. Bu nedenle takip edilen bir geliştirme standardı da bulunmamaktadır. Dolayısıyla uzaktan eğitim sistemlerinde yazılım hata risk oranı artmaktadır.

Uzaktan eğitimdeki öğrenci sayısı artması ve yazılım geliştirme standartlarına uyulmaması yazılım bakım maliyetlerini arttırmaktadır.

1.2. Yapılan Çalışmalar

ODTÜ'de NET-Class isimli yazılım geliştirilmiştir [2]. 3 rol üzerinden modüller tanımlanmıştır. Sakarya Üniversitesi Enformatik bölümünde internet destekli öğretim projesi geliştirilmiştir [3]. Öncelikle ön lisans programlarında uzaktan eğitim uygulanmıştır. Anadolu Üniversitesi Açık Öğretim Fakültesi'nde bazı dersler (genel matematik gibi) için uzaktan eğitim altyapısı kullanmaktadır. Bu sistemler tasarlanırken açık kaynak kodlu Moodle, e-LMS, OSL-Learning gibi uzaktan eğitim yazılımlarından faydalanılmıştır.

Geliştirilen sistemlerde göz önünde bulundurulan özellikleri aşağıdaki gibi sıralayabiliriz [4]:

a. Kolay ve hızlı materyal erişimi,
b. Etkin rol tanımı,
c. Görsel eğitim dokümanlarına ağırlık verilmesi,
d. Bölüm sonu ölçme değerlendirme,
e. Başarı oranının ölçülmesi,
f. Kullanım kolaylığı,
g. Bilgi güvenliği.

Açık kaynak kodlu yazılımlarda yukarıdaki maddelerden bilgi güvenliği sağlanmasında eksiklikler göze çarpmaktadır.

Sadece ders materyaline erişim sağlayan uzaktan eğitim sistemleri yetersizdir. Kullanım kolaylığından uzak, güvenlik açıkları olan uzaktan eğitim sistemleri etkili ve verimli kullanılamamaktadır. Yazılım standartlarına uymayan bir uzaktan eğitim sistemi öğrenim ölçme ve değerlendirme standartlarına da uymamaktadır. Hem yazılımsal hem de öğretim hataları olan bir uzaktan eğitim sistemi amacına ulaşmamaktadır.

Bu eksikliklerin giderilebilmesi için yazılım geliştirme sürecinde yazılım test adımlarına uymak gerekmektedir.

1.3. Tezin Amacı

Bu çalışmada yazılım test teknikleri kullanılarak uzaktan eğitimde ÖDS geliştirilmiştir. ÖDS yazılımı yazılım mühendisliği temel yaşam döngülerinden v-model adımlarına uyularak hazırlanmıştır. Her yazılan birim yazılım test teknikleri ile test edilmiş ve yeni bir ölçme değerlendirme sistemi önerisi getirilmiştir. Yazılım test süreci, kod karmaşıklığı fazla olan, güvenlik açığı en aza indirilmiş, bakım maliyetleri düşük bir sistem isteniyorsa mutlaka her yazılım sistemine uygulanmalıdır. Bu çalışmada uzaktan eğitimde ÖDS'ye yazılım test teknikleri uygulanmış ve elde edilen sonuçlar grafiklerle ifade edilmiştir. Yapılan çalışmanın hem yazılım test mühendisliğine hem de yapılacak uzaktan eğitim çalışmalarına model olması hedeflenmektedir.

1.4. Önerilen Yöntem

Bu çalışmada uzaktan eğitimin tarihçesi ve yapılan uzaktan eğitim çalışmalarına değinilmiş, yazılım test tekniklerinin kullanım amaçları açıklanmış, uzaktan eğitimde bir ÖDS tasarlanmış ve yazılım test teknikleri ile bu sistem test edilmiştir. Elde edilen sonuçlar ve öneriler belirtilmiştir. Tasarlanan sistem için Visual Studio 2010 Ultimate aracı ve .Net Framework 4.0 kullanılmıştır. Veri tabanı SQL Server 2008 ile tasarlanmıştır. ÖDS için tasarlanan web sayfaları Selenium IDE, Vega Subgraph, Visual Studio PerformanceAnalyzer ile çeşitli testler ile test edilmiş elde edilen sonuçlara göre sistem yeniden kodlanmıştır. Veri tabanı performans analizi için SQLMon test aracı kullanılmıştır.

1.5. Tez Planı

Bölüm 1'de tezin problem tanımı yapılmış, tez amacı açıklanmış ve tezde hangi yazılım araçlarının kullanılacağı belirtilmiştir. ÖDS ile ilgili daha önce yapılan çalışmalar hakkında bilgi verilmiştir. ÖDS sisteminin hangi yazılım test araçları ile test edileceği de Bölüm 1'de belirtilmiştir. Bölüm 2'de yazılım testi açıklanmış, yazılım süreç geliştirme modellerinden ve ÖDS'de hangi modelin kullanılacağından bahsedilmiştir. Müteakiben yazılım test tekniklerine değinilmiş ve ÖDS'de hangi yazılım test tekniklerinin kullanıldığı açıklanmıştır. Bölüm 3'te uzaktan eğitim hakkında genel bilgilendirme yapılmış, uzaktan eğitimin

avantajlarından ve kullanım alanlarından bahsedilmiştir. Bölüm 4'te ÖDS tasarım adımları açıklanmış ve geliştirilen sistemdeki web sayfaları ve kullanım amaçları açıklanmıştır. Bölüm 5'te ÖDS yazılım test araçları ile test edilmiş ve elde edilen sonuçlar açıklanmıştır. Bölüm 6'da tezin genel sonuçları ve bilime kattığı değer ele alınmıştır.

BÖLÜM 2. YAZILIM TESTİ

2.1. Yazılım Testi Nedir?

Yazılım testleri, geliştirilen bir yazılım sisteminin kontrolünün yapılarak ihtiyaçların ve gereksinimlerin tam olarak karşılanması için var olan işlemler bütünüdür [5].

Boch, yazılım testini, "korunmak için, akla gelmeyecek kadar gizli-kapalı belirsizlikleri karşılaştırma sürecidir." şeklinde tanımlamıştır. Test etme, aslında bir süreçtir. Amacı, bir sistemdeki hataları ve eksiklikleri bulmaktır. R. Vaderwall' göre, "Yazılım testi, ürünün davranışlarını öngörme/tahmin etme ve bu tahminlerin gerçek sonuçlarla karşılaştırılması sürecidir." Test sürecinin projeye ilk evrelerde entegre edilmesi maliyeti azaltır ve hata bulma şansını arttırır [6].

Literatürdeki farklı yazılım test tanımlarından bazıları şunlardır [7]:
- Yazılım testi, bir programın davranışının beklenen davranışa uymadığı durumları bulma işlemidir [7, 8].
- Bir sistemin veya bileşenin belirli koşullar altında çalıştırılması, sonuçların gözlenmesi, kaydedilmesi ve belirli özelliklerin değerlendirilmesi sürecidir [7, 9].
- Test, hata bulma amaçlı planlı bir şekilde gerçekleştirilen eylemler dizisi, bir doğrulama yöntemidir [7, 10].
- Bir yazılım öğesinin mevcut ve olması gereken koşullar arasındaki farkın bulunarak analiz edilmesi sürecidir [7, 11].
- Test bir yazılım ürününün zayıf yönlerini veya makul hatalarını keşfetmek için gerçekleştirilen bir süreçtir [7, 12].

Testler, sadece test durumlarının iyi olmasını değil, aynı zamanda test edilecek tüm gereksinimlerin de iyi olmasını araştırır. Bununla birlikte, hataların tamamı test esnasında bulunamayabilir. Yazılım testinin önemi, olayların zamanında test

edilmemesi ve kalitesiz bir ürün elde edilmesiyle ortaya çıkar [13]. Yazılım testi, bir yazılımın test durumları kullanılarak dinamik doğrulanmasıdır [14].

2.2. Yazılım Testi Kullanım Alanları

a. Finans
b. Telekominikasyon
c. Ar-ge
d. Savuma Sanayi
e. Bilgi Sistemleri

Testler, bir ürünün hazır olup olmadığını denemek için kullanılmasına rağmen diğer amacı da, hazır olmayan ancak doğru bir şekilde çalışan ürünü de sınamaktır. Bir başka görüşe göre testin amacı, yazılımın standartlara ve kabul şartnamesine uygunluğunu tespit etmek, yazılım geliştirme sürecine katkı sağlamak ve yazılım kalite güvencesi aktivitelerini uygulamaktır. Bunlara ilaveten, yazılım hataları ve yazılımın çalışma sistemine dair riskleri yönetilebilir kılmak, kodun ileriye dönük geliştirilme masraflarını azaltmak, ürün çalıştırılmadan önce kalitesini ve senaryolara uygunluğunu denetlemek, geliştirme sırasında gözden kaçan yanlışları bulmak ve bu yanlışların ileride de tekrarlanmasını önleyerek zaman ve maliyet tasarrufu yapmaktır. Özetle, kalite veya kabul edilebilirlik hakkında karar vermek ve problemleri keşfetmektir [15].

Bu kapsamda dikkat edilmesi gereken hususlar şunlardır [16]:

a. Dinamizm: Yazılım mutlaka çalıştırılarak test edilmelidir.
b. Sınırlılık: Yazılımın neredeyse sonsuz sayıda olabilecek çalışma alanlarının tümünün testi imkânsız olacağından, kritiklik düzeylerine göre sıralanıp yeterli görülen sayıda en kritikleri test edilmelidir.
c. Uygunluk: Test edilecek davranışın doğasına uygun ve muhtemel riskleri göz önünde bulunduran testlerin gerçekleştirilmesidir.

Yazılım testinde beklenen davranışlar, test edilecek yazılımın, kullanıcı beklentilerine, gereksinimlerine ve akla uygun, mantıklı beklentilere cevap verebildiğinin test edilmesidir [17].

Muller ve arkadaşları ise yazılım testinin amaçlarını şu şekilde özetlemektedir [18];

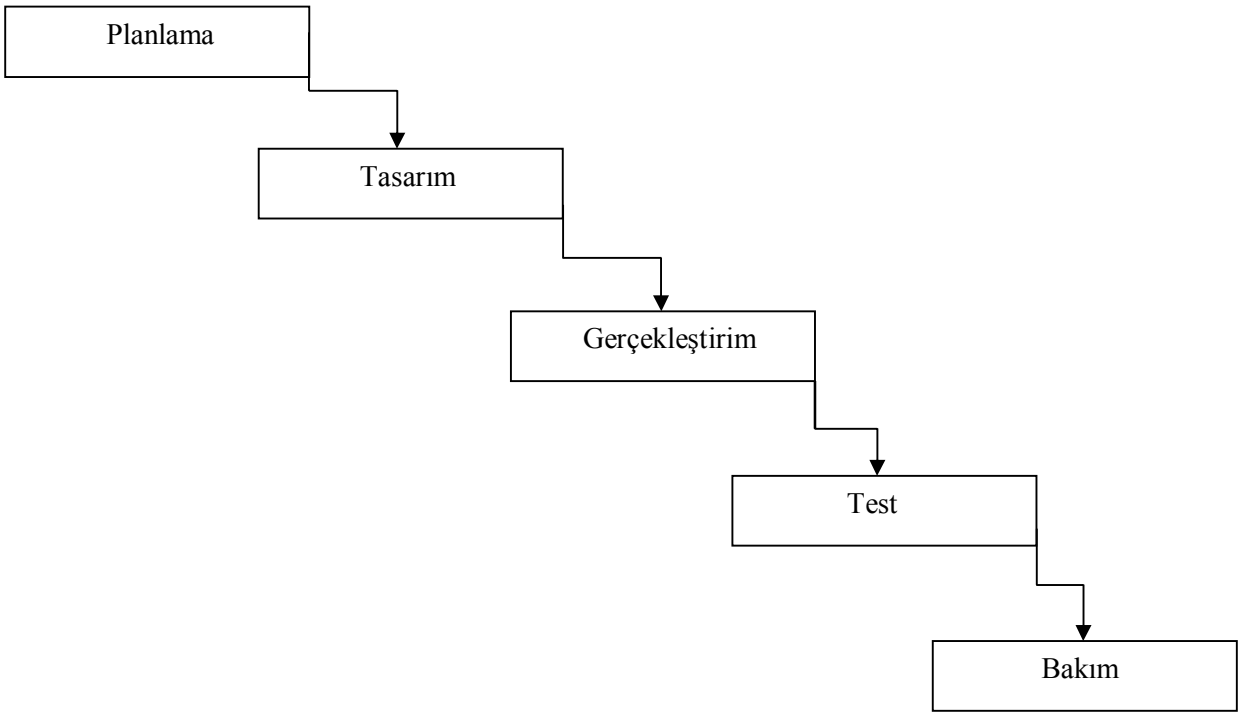
- Bir kişiye, bir ortama veya bir şirkete zarar verebilecek yazılım hatalarını belirlemek.
- Hataların ana sebepleri ve etkileri arasındaki farkı ayırt edebilmek.
- Elde edilen örnekler yardımıyla testin niçin gerekli olduğunu tayin edebilmek.
- Test etmenin niçin kalite güvencenin bir parçası olduğunu ve elde edilen örneklere bakarak daha yüksek kalite sağlamak için nasıl bir testin olması gerektiğini saptamak.
-Hata(error), kusur(defect), aksaklık (fault), bozukluk(failure), yanlışlık(mistake) ve yanlış(bug) gibi terimleri yeniden hatırlamak.
- Yazılım projesinin başlangıcından bitmesine kadar olan süreçte hataları en aza indirebilmek.
- Projenin teslim edildikten sonraki teknik desteğini kolaylıkla yapabilmek, yeni gereksinimleri projeye daha rahat entegre edebilmek ve kullanım kolaylığı gibi konularda eksik olmamak.

2.3. Test İşleminin Yazılım Geliştirme Sürecindeki Önemi

Yazılım hacminin, geliştirme gruplarının ve başarısız projelerin artmasıyla, projelerin başarısını arttırmak için geliştirme süreçleri tanımlanmaya başladı.
Projelerin karmaşıklığının artmasıyla, havaalanı kontrolü ve uçuş rezervasyon sistemleri gibi, nesne yönelimli programlama ve gerçek zamanlı işletim sistemleri gelişmiştir [6]. Yazılım geliştirme süreci bir yazılım ürününün geliştirilmesi ile ilgili bir grup eylemdir [7, 19]. Geliştirme süreci, temel olarak, yazılımın geliştirilmesini veya değerlendirilmesini amaçlayan faaliyetler kümesidir ve yazılım sürecindeki genel faaliyetler; tanımlama (specification), geliştirme (development), onaylama (validation) ve değerlendirme (evolution) aşamalarından oluşur [6].

Yazılım mühendisliği, iyi tanımlanmış süreç içerisinde geliştiriciye kılavuzluk yapmak ve doğru yazılımın geliştirilmesini sağlamak için vardır. Yaşam döngüsü, yazılım geliştirme sürecini ve yazılımın bakım-onarımını organize etmek için kullanılır. Nihai amaç, makul bir fiyatta kaliteli yazılım ürünü üretmektir [20].

Eksiksiz bir proje yürütmek için uygulanması gereken yazılım geliştirme süreci Şekil 2.3'te gösterilmektedir.



Şekil 2.3. Şelale yazılım geliştirme süreç modeli

Aşağıda Şekil 2.3'te gösterilen aşamaların tanımı yapılmaktadır.

-Planlama; üstesinden gelindiği düşünülen problemin çözümünü tanımlar. Bu kısımda; tam, kesin, açık ve anlaşılabilir biçimde problemin gereksinimlerini içeren çeşitli gereksinimler bulunur.

-Tasarım; sistem modelinin bileşenlerinin detaylı yapısı ve davranışlarını tanımlar. Bu kısımda, modelin her bir bileşeninde bulunan veri yapıları ve algoritmalar detaylı bir şekilde tanımlanır.

-Gerçekleştirim; programlama dilindeki tasarım çözümünün değişimini içerir. Bu, uygulamadaki programların gerçek yapısıdır. Bu bölüm, oluşturulmuş programlar kümesidir.

-Test; elde edilen örnek kümesine ve uygun veriyle gereksinimlere göre programların çalışıp çalışmadığını doğrular.

-Bakım; kullanıcı için programın kurulumunu, teslimini ve sonuçların değerlendirilmesini kapsar.

Bu sürecin tamamından ortaya çıkan sonuç, uygun verilerle bilgisayar ortamında test edilen ve makul sonuçlar elde edilen iyi belgelenmiş bir bilgisayar programıdır. Yazılım geliştirme süreci, yazılım yaşam döngüsünün bir parçası olarak düşünülebilir. Yazılım geliştirme süreci, iyi bir program elde etmek için takip edilecek görevlerin iyi tanımlı bir sırasıdır [20].

Yazılım geliştirme sürecinde asıl hedef, tasarımda iyi iş yapmayı, testte iyi iş yapmaya tercih etmek (hata ayıklayıcı yerine tasarım aracı kullanmak), kaliteyi hataları düzelterek değil önleyerek sağlamak, bakımları da hata önleyici bakım olarak görmek, tasarım ile gereksinimleri eş zamanlı olarak güncellemek, sürece odaklanmayı etkinliklere tek tek odaklanmaya tercih etmek, çapraz-fonksiyonel (cross-functional) ekipler kurmaktan ve bu ekiplere müşteriyi katmaktan çekinmemek ve sorumluluğu tüm ekibe dağıtmaktır [21].

2.4. Yazılım Test Prensipleri

Günümüze ait yazılımlarda değişik hatalar vardır ve bu hatalar bazen bilgi kayıplarına ve değişmelere sebep olur. Kalite, piyasaya sürülen yazılımlardaki

hataları azaltmak için yazılım testi tarafından değerlendirilir. Geleneksel olarak yazılım testi, yazılımdaki mümkün olabilecek hataları bulmayı aramaktan ziyade, yazılımın karakteristiklerini ortaya çıkarmaya çalışır. Kaliteli bir yazılım ürünü ortaya çıkarabilmek için uyulması gereken bazı prensipler bulunmaktadır. Bu prensipler şunlardır:

- Test işlemi, bağımsız gruplar tarafından yapılmalı,
- Test için en iyi personel seçilmeli,
- Test, maksimum hata sayısı elde etme amacıyla planlanmalı,
- Geçersiz ve beklenmeyen giriş durumları, geçerli durumlar kadar iyi test edilmeli,
- Test esnasında test altındaki yazılımda değişiklik yapılmamalı,
- Test durumları ve test sonuçlarını içeren test raporu hazırlanmalı,
- Mümkünse, beklenen sonuçlar belirlenmeli ve rapora dahil edilmeli,
- Test ilerledikçe planlanmalı ve daha sonra güncellenmeli ve
- Uygun bir test metodu seçilmelidir [22].

2.5. Yazılım Test Planı

Yazılım projelerinde proje başarısını doğrudan etkileyen eylemlerin başında iyi bir test planlaması gerekmektedir. Bu amaçla yazılım projelerinin farklı aşamalarında test planları hazırlanır. Bu planlarda test edilecek yazılım parçaları, özellikler (işlevsellik, performans, güvenlik, kullanılırlık vb.), icra edilecek görevler, çıktılar, gerekli kaynaklar, sorumluluklar, takvim ve gerekli onaylar tanımlanır. Aynı zamanda projelerde test eylemlerini daha doğru ve etkin tanımlamak için farklı seviyelerde birden fazla test planı üretilebilir. Örneğin, projenin en genel test yaklaşımını belirten Test Ana Planı, sistem testleri yaklaşımını belirten Sistem Test Planı veya yazılım testleri yaklaşımını belirten Yazılım Test Planı örnek olarak verilebilir. Tüm bu planlar IEEE 829-1998 standartları kapsamında belirtilen test planı şablonuna göre geliştirildiğinde aşağıdaki başlıklar ortaya çıkar [7]:

a. Test plan tanımlayıcısı
b. Giriş

c. Test öğeleri
d. Test edilecek özellikler
e. Test edilmeyecek özellikler
f. Yaklaşım (Strateji)
g. Geçme / Kalma kriterleri
h. Testi durdurma ve teste yeniden devam etme kriterleri
i. Test çıktıları
j. Test görevleri
k. Çevresel gereksinimler
l. Sorumluluklar
m. Personel ve eğitim gereksinimleri
n. Takvim
o. Riskler ve öngörülmeyenler
p. Onaylar.

Bir test planı, test aktiviteleri ve hizmetleri gibi yaklaşımları tanımlayan bir belgedir. Test planı, gelişim döngüsünün ilk evrelerinde hazırlanmalı ve analiz, tasarım ve kodlama aktivitelerinin etkileşimlerini geliştirmeye yardımcı olmalıdır. Test planı; testin amaçlarını, alanını, stratejisini ve yaklaşımını, test prosedürlerini, test ortamını, test tamamlama kriterlerini, test durumlarını, test edilecek bileşenleri, uygulanacak testleri, test zaman çizelgesini, personel gereksinimlerini, raporlama prosedürlerini, tahminleri, riskleri ve olasılık planlamasını belirler. Bir test planı geliştirilirken, test planının uygun bireyler tarafından erişilebilir ve hazır olduğundan emin olunmalıdır. Test planı hazırlamanın iki yöntemi vardır. İlk yaklaşım, her bir test planının detaylı bir şekilde gösteren ana test planıdır. Detaylı test planı, şelale geliştirme yaşam döngüsündeki belirli evreleri takip eder. Test planı örnekleri; sınıf/birim, entegrasyon, sistem ve kabul testlerini kapsar. Diğer detaylı test planları, uygulama yükseltmelerini, gerileme testini ve paket kurulumlarını kapsar. Birim test planları, kod merkezlidir ve çok detaylıdır. Fakat, sınırlı alanları olduğu için kısadır. Sistem veya kabul test planları, tam olarak yazılım birimi değil de tüm sistemin kara kutu görünümünü veya fonksiyonel testi üzerine odaklanır. İkinci yaklaşım, bir test planıdır. Bu yaklaşım, sık kabul/sistem test planı olarak isimlendirilir fakat birim,

entegrasyon, sistem ve kabul testi ve testlerin tamamlanması için planlama düşüncelerinin hepsini içeren bir test planıdır. Bir test planı, adım tüm süreci tanımlar ve test uygulanır. Bu plan, testin amaçlarını ve koşullarını test etmek için gerekli adımları, veri girdilerini, beklenen sonuçları ve gerçek sonuçları kapsar.

Yazılım, ortam, sürüm, test ID, ekran ve test tipleri gibi diğer bilgileri de içerir [23]. İyi bir test planı hazırlamak için yapılması gereken adımlar şunlardır;

a. Test amaçlarını belirlemek
b. Test tekniği geliştirmek
c. Test ortamı belirlemek
d. Test özelliklerini geliştirmek
e. Testi planlamak
f. Testi gözden geçirmek ve onaylamak**.**

Test planı, yazılımın yüksek kalitede olup olmadığını, herkes tarafından anlaşılıp anlaşılmadığını ve sorumluluklarını yerine getirip getirmediğini tespit etmek için hazırlanır. Test planı, proje üzerinde kontrollü bir şekilde tekrar düzeltme yapılabilir [23].

Etkili bir test, iyi bir planlama ve uygulamayı gerektirir. Test planı;

a. yapılacak olan testin hedefini belirlemeyi,
b. zaman tahmini, kaynakları, personeli, donanımı, yazılımı ve araçları belirlemeyi
c. ihtiyaç duyulan kaynakları sağlamayı,
d. test ortamını sağlamayı,
e. görevlere personel tayin etmeyi,
f. planı belirlemeyi,
g. risk ve olasılık planlarını tanımlamayı,
h. süreç takibini ve doğru adımlar atmayı,
i. geçilen, engellenen ve başarısız testlerin düzenli test durumlarını sağlamayı,
j. projenin hedefi değişirse yeniden plan yapmayı,

k. herhangi bir bölümü anlamak için son durum incelemesi yapmayı içerir [23].

Test proje planının amacı, belirli bir olayda başarılı bir test gerçekleştirmek için kural oluşturmaktır. Burada en önemlisi belgelemedir. Çünkü belgeler, test projesini yönetmeye yardım eder. Eğer bir test planı yeterince kapsamlıysa ve dikkatlice düşünülmüşse, test uygulama/çalıştırma ve analizi de düzgün bir şekilde yürümelidir. Test proje planı, özellikle spiral ortamda sistem sürekli değiştiği için gelişen bir belgedir. İyi bir test planı için aşağıdaki tespitler yapılabilir;

a. hata tespit şansı yüksektir,
b. test kapsamı geniştir,
c. esnektir,
d. kolay ve otomatik bir şekilde çalıştırılır ve tekrarlanabilir,
e. uygulanacak test tiplerini belirler,
f. sonuçlar belgelenir,
g. hata bulunduğunda hatayı düzeltme imkânı sağlar,
h. test amaçlarını açıkça tanımlar,
i. test stratejileri belirtilir,
j. test çıkış kriterlerini açıkça tanımlar,
k. riskleri tanımlar,
l. test gereksinimlerini belirler,
m. testin teslim edilebilirliğini belirler [13].

2.6. Yazılım Test Tasarımı

Test planlama süreci başarılı bir şekilde tamamlandıktan sonra "Test Tasarımı" evresi başlar. Test tasarımı, öncelikle belirli test gereksinimlerini içerir. Test etme, test tasarım mühendislerine test verilerinin uygunluğunun doğrulanmasına izin verir ve test verilerinin düzenini sağlar. Testleri ve test verilerini tasarlama, test sürecinin çok fazla zaman alan kısmıdır. Aktivitelerin ayarlanması da oldukça önemlidir. Eğer testler, gereksinimleri karşılamazsa, testler geçersiz sayılır. Test verileri, testlerin amacını yansıtmazsa, o testler de geçersizdir [23]. Test tasarım süreci; test ortamının

hazırlanması, test durumlarının yazılması ve test yordamlarının hazırlanması faaliyetlerinin icra edilmesiyle tamamlanır [13].

Test planı geliştiricileri fiziksel test imkânlarını, donanımı, yazılımı, ağları, gerekli teknik desteği, test için gerekli olan özel yazılım yerlerini denetlerler ve söz konusu öğeler için bir plan hazırlarlar. Test ortamının amacı, test aktiviteleri için gerekli fiziksel ortamı sağlamaktır. Buna göre, test ortamının ihtiyaçları belirlenir ve uygulamaya başlamadan önce tekrar gözden geçirilir. Test ortamının içerdiği ana bileşenler, fiziksel test imkânı, teknolojiler ve araçlardır. Test imkânı bileşeni; fiziksel durumu içerir. Teknolojiler bileşeni; donanım platformları, fiziksel ağ ve onun tüm bileşenleri, işletim sistemi yazılımı ve yardımcı yazılımlar gibi diğer yazılımları içerir. Araçlar bileşeni ise otomasyon test araçları, test etme kütüphaneleri ve yazılım desteği gibi herhangi bir özel test yazılımını içerir. Test ortamı ve çalışma alanının kurulması gerekir. Bu, bireysel çalışma alanından resmi test laboratuarlarına kadar çeşitlilik gösterir. Test ediciler ve geliştirme takımları arasında her olayda bir yakınlık olması önemlidir. Bu iletişim kolaylığı ve ortak amaçta buluşma kolaylığı sağlar. Elde edilen test araçları, donanım ve yazılım teknolojilerinin kurulması gerekir. Bu, test donanımının ve yazılımının kurulumunu, üretici, kullanıcı ve bilgi teknolojileri personeliyle koordinasyonu kapsar. İletişim ağlarının da kurulması ve test edilmesi gerekir [13].

Test ortamı, organizasyon, iş ve proje gereksinimlerine göre değişir. Yüksek kalitenin kısıtlı olduğu önemli alanlarda, laboratuar mühendislerine tahsis edilen test laboratuarı kullanılabilir ve yazılım test ediciler tarafından laboratuar zamanının belirlenmesi gerekebilir. Küçük bir proje için küçük organizasyonlardaki bir çalışma yeri test ortamı olarak yeterli olabilir. Test ortamı, yazılımın hatasız olduğunu doğrulamada projeyi desteklemek içindir ve bir test ortamı belirlemek için önemli derecede maddi yatırım gerekebilir. Test ortamı, yazılımın doğruluğunu onaylamak için gerekli donanım ve yazılımı içerir [11].

Test durumu, belirli bir program parçasının çalıştığının veya bir gereksinimin doğrulandığının gösterilmesi için kullanılan girdiler, gerçekleştirilmesi gereken

adımlar ve beklenen sonuçların belirtilmesidir [11]. Bir test durumu adım adım bir testin nasıl icra edileceğini tanımlar. Bir test durumunda olması gerekenler [11];

a. Testin durumunun amacı ve gerçekleştirilme şartı,
b. Test durumu ile ilgili test ortamının adım adım kurulması,
c. Girdi verileri,
d. Beklenen sonuç,
e. Gerçekleşen sonuç,
f. Yazılımın sürüm tanımı,
g. Yazılımın çalışma ortamı,
h. Test ID'sidir .

Bir test durumu yazılıma sorulan bir sorudur ve test edilen öğenin sadece bir özelliğini test etmelidir [7].

Test yordamı, her bir test durumunun test ortamının kurulması, koşturulması ve sonuçlarının değerlendirilmesi için ayrıntılı direktifler, açıklamalar listesi içeren ve test planı temel alınarak geliştirilen belgedir [7, 9]. Diğer bir ifade ile test yordamı "bir ya da birden fazla test durumunun koşturulması için hazırlanan detaylı açıklamaları içeren belgedir". Her bir test durumu için ayrı test yordamları olabileceği gibi bir grup halinde olan test durumları için de bir test yordamı hazırlanabilir [7].

2.7. Yazılım Test Çalıştırma

Test çalıştırma, ilk olarak test amaçlarının karşılanmasıyla başlar ve test etmek için tüm kriterler uygulanır. Test çalıştırma esnasında testi yaparken birçok sayıda gözlemlenen test senaryoları ve davranışları sağlanmasına rağmen testler, test prosedürlerine göre çalıştırılmalıdır. Test çalıştırma aktivitelerinin merkezinde beklenen sonuçlarla gerçek sonuçlar bulunur. Test ediciler çok dikkatli olmalı ve bu görevlere odaklanmalıdırlar, aksi takdirde, hatalar bulunmadığında veya doğru olması gereken davranışlar yanlış bulunduğunda, testi tasarlama ve uygulama

işlerinin tamamı boşa yapılmış olabilir. Eğer, beklenen ve gerçek sonuçlar hesaplanmazsa, hata olayı meydana gelir. Olaylar, (test hedefinde bir hata olsun veya olmasın) sebepleri tespit etmek ve olayların sonuçlarıyla veri toplamak için dikkatle irdelenmelidir. Bir hata belirlendiğinde, test özelliklerinin doğru olduğunu tespit etmek için dikkatlice değerlendirilmelidir. Hazırlanan bir testin test durumları, test kodu veya çalıştırılma şekli gibi konularda yanlışlıklar olabilir. Test temelinde ve test amacında değişiklikler olduğu için, çoğu kez başarılı bir şekilde çalışsa bile test özelliği yanlış olabilir. Test ediciler, gözlenen sonuçların yanlış test sonuçları olabileceği durumunu da düşünmelidirler. Test çalıştırma esnasında, test sonuçları, uygun bir şekilde kaydedilmelidir. Çalışan fakat sonuçları kaydedilmeyen testler, etkisizliğini ve gecikmeleri göstererek doğru sonuçları belirlemek için tekrarlanmak zorundadırlar.

Test amaçları, test araçları ve test ortamlarının tamamı değerlendirildiği ve kaydedildiği için test edilmiş özel sürümler belirlenmelidir. Test kaydetme, testin çalışmasıyla ilgili detayların kronolojik kayıtlarını sağlar. Sonuçları kaydetme, hem bireysel testleri hem de olayları kapsar [18].

Test uygulama ve çalıştırmayı izlemek için metrikler;
- test ortamlarının ayarlanma yüzdesini,
- test veri kayıtlarının yüklenme yüzdesini,
- test durumları ve vakaların çalıştırılma yüzdesini ve
- test vakaların otomatikleştirilme yüzdesini kapsar [18].

2.8. Yazılım Test Değerlendirme

Öncelikle ölçütler analiz edilmelidir. Ölçütler, daha etkili kararlar verebilmek ve geliştirme sürecini desteklemek için kullanılır. Bunun amacı, test sürecini kontrol etmek için metrik prensiplerini uygulamaktır. Daha sonra, proje durum raporu, test hata detayları raporu ve hata raporu gibi raporlar hazırlanarak ara rapor yayınlanır [13]. Test çerçeve planı çıkartılır. Geliştirme esnasında test çerçeve planının sürekli izlenmesi gerekir. Amaç, son durumu göstermek için test planını güncellemektir. Son

olarak, gereksinim değişiklikleri belirlenir. İlk gereksinim belirlemede, işlevsel kompozisyon, fonksiyonel pencere yapısı, pencere standartları ve sistem gereksinimleri gibi test fonksiyonları analiz edilir. Değişen yeni gereksinimler ise aşağıdaki konuları içerebilir;

a. kullanıcı arayüzleri ve bileşenleri,
b. yeni fonksiyonlar,
c. değiştirilmiş fonksiyonlar,
d. elenmiş fonksiyonlar,
e. yeni sistem gereksinimleri, ve donanım gibi,
f. ilave sistem gereksinimleri,
g. ilave kabul gereksinimleri.

Her bir yeni gereksinimin test planında, test tasarımında ve test dilinde tanımlanması, kaydedilmesi, analiz edilmesi ve güncellenmesi gerekir [13].

2.9. Yazılım Test Teknikleri

2.9.1. Statik test

Detaylı bir yazılım testi değildir. Algoritma, doküman ve kodla ilgilenir. Kod yazım ve kod özelliklerindeki hataları bulmaya çalışır. Bu tipteki test kodu yazan geliştirici tarafından yapılır. Yazılım gereksinim ve özelliklerinin hedeflenen sistemle uygunluğunun test edilmesidir.

Statik testte kod üzerinden kontrol yapılırken önceden hazırlanmış bir kontrol listesi üzerinden kontrol yapılır. Şekil 2.4'te statik teste ait kontrol listesi görülmektedir.

Güvenilirlik	Perfomans	Kullanılabilirlik
Güvenilirlik gereksinimleri belirtilmiş mi?	Cevap ve gecikme zamanı gereksinimi belirtilmiş mi?	Kullanılabilirlik gereksinimi belirtilmiş mi?
Sağlamlık gereksinimleri belirtilmiş mi?	Çıktı gereksinimleri belirtilmiş mi?	Renk şemaları ve standartlar uygun mu?
Servis gereksinimleri belirtilmiş mi?	Veri hacmi gereksinimi belirtilmiş mi?	

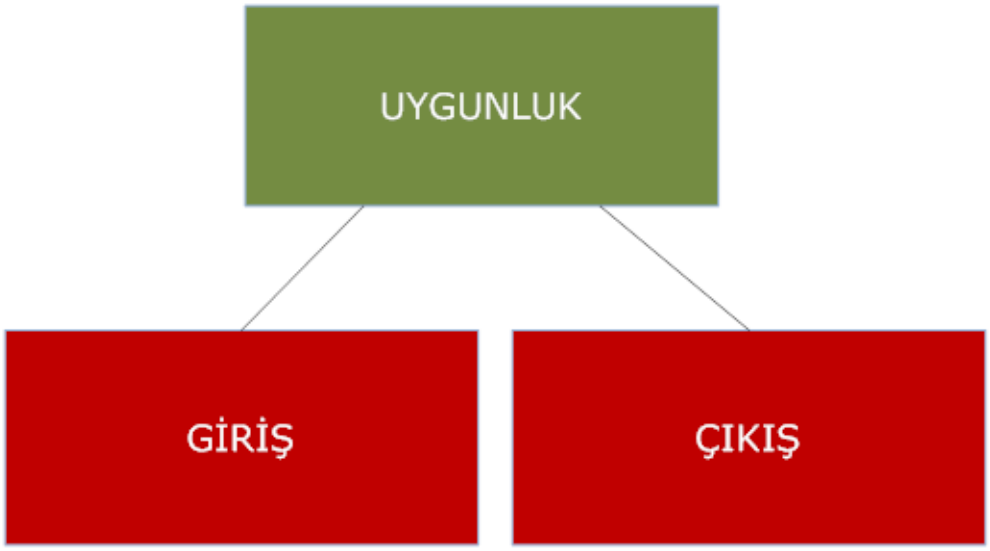
Şekil 2.4. Statik test kontrol listesi

Statik testlerde yazılımın doğruluğu kontrol edilir. Bu amaçla kodun çalıştırılması yerine yazılan kod, hata bulmak amacıyla okunur. Bu okumada kod gözden geçirilerek incelenir ve statik olarak analiz edilir [7]. Statik testlerin önemi, hataların yazılım yaşam döngüsü içerisinde erken safhalarda yakalanmasını sağlamasıdır [24].

2.9.2. Dinamik test

Dinamik testte kodun yapısal analizi ile ilgilenilmez. Bunun yerine yazılımın davranışı üzerinde durulur. Bunun için sisteme belli giriş değerleri verilip beklenen çıkış değerleri gözlenir.

Geliştirilen yazılımın dinamik davranışının gözlemlenmesi için gerçekleştirilen testlerdir. Dinamik testler kapsamında sistemin değişen veriler karşısında nasıl tepki verdiği gözlemlenir [7]. Dinamik testler, fonksiyonel testlerdir. Kara kutu testi, beyaz kutu testi, gri kutu testi ve tecrübeye dayalı test tekniği olmak üzere kendi içerisinde maddelere ayrılır. Şekil 2.5'te bu tekniklerin tümü kısaca açıklanmaktadır.



Şekil 2.5. Kara kutu uygunluk grafiği

Dinamik test tipi olarak kara kutu test biçimini örnek gösterebiliriz.

2.9.2.1. Dinamik test tipleri

Birim testi:

Birim Test yazılımın en küçük test edilebilir parçasının test edilmesi işlemidir. Bu en küçük birim program, fonksiyon, prosedür ve sınıf olabilir. Birim test işlemini geliştirici yapar. En alt seviye testidir.

Aşağıdaki C# kod parçasında bir test sınıfı ile toplama işleminin test edilme işlemi yapılmaktadır.

```
public class TestToplam {
   public void testTopla () {
      Toplayıcı islem1= new Uygulama();
      // pozitif değerler toplanabilir mi?
      assert(islem1.topla(1, 1) == 2);
      assert(islem1. topla (1, 2) == 3);
      assert(islem1. topla (2, 2) == 4);
```

```
        // Sıfır değeri?
        assert(islem1. topla (0, 0) == 0);
        // negatif değerler toplanabilir mi?
        assert(islem1. topla (-1, -2) == -3);
        // pozitif ve negatif değerler toplanabilir mi?
        assert(islem1. topla (-1, 1) == 0);
        // büyük değerler toplanabilir mi?
        assert(islem1. topla (1234, 988) == 2222);
    }
}
interface Toplayıcı{
    int Topla(int a, int b);
}
class Uygulama implements Toplayıcı {
    int Topla(int a, int b) {
        return a + b;
    }
}
```

Birim testinin altı kuralları aşağıda listelenmektedir:

a. Testi yazma işlemi,
b. İlk başarılı test işleminden sonra yazma işini bırak,
c. Çalışmayan bir durumla başla,
d. Test edilebilirliği devam ettir ve
e. Sahte nesne kullan.

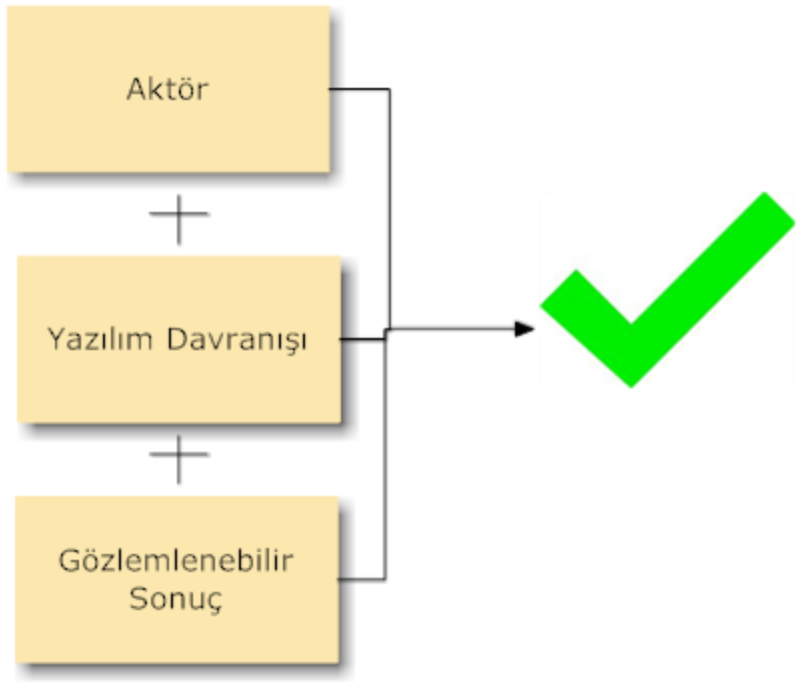
Bütünleşme testi:

Bütünleşme testi birim testten sonra uygulanan bir aşamadır. Sistemin yazılım modüllerinin uyumluluğu test edilir. Burada üç tip yaklaşım uygulanır:

- Yukarı-Aşağı: En büyük modülden alt modüllere doğru uyumluluk tespiti.
- Aşağı-Yukarı: En küçük modülden en büyük modüle doğru uyumluluk tespiti.
- Katmanlı: Bir en üst modül ile bir en alt modülün uyumluluğunun test edilmesi.

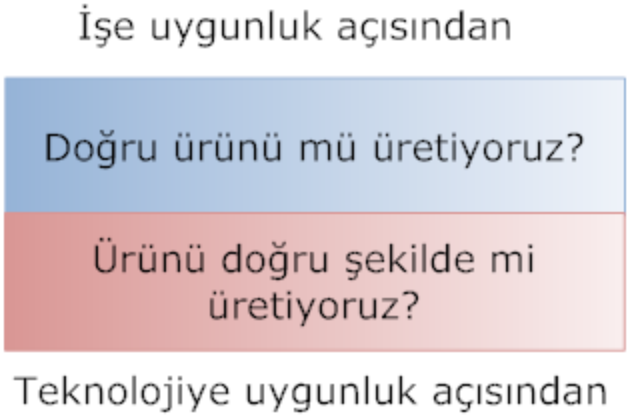
Kabul testi:

Geliştirilen yazılımın kullanıcı gereksinimlerini karşılayıp karşılamadığı test edilir. Testin son aşamasıdır. Kabul testi son kullanıcı tarafından yapılır. Kabul testi kriterleri şeması Şekil 2.6'da görülmektedir:



Şekil 2.6. Kabul testi akışı

Aktör, kullanılacak yazılımı kullanır ve yazılım işlevlerini inceler. Eğer yazılım hedeflenen şekilde davranış gösteriyorsa sistemi onaylar. Bu işlemler sırasında belli sorular sisteme sorulur. Kabul testi soruları ise Şekil 2.7'de listelenmektedir:



Şekil 2.7. Kabul testi soru tablosu

Kabul Testi örneği:

Bir butona basılma işleminde kabul testi için aşağıdaki sorular sorulmalıdır:

a. Butona hangi sıklıkla basılmalı?
b. Butona ne zaman basılmalı?
c. Butona kim basmalı?

Kabul testi araçlarından bazıları aşağıda listelenmiştir [23]:

1. FitNesse, Framework for Integrated Test (Fit)
2. iMacros
3. ItsNat Ranorex
4. Selenium (software)
5. Test Automation FX
6. Watir
7. Fabasoft app.test

Kara kutu testi:

Programın giriş ve çıkış değerleri ile ilgilenir, kodlama ve iç yapısıyla ilgilenmez. Kara kutu işlem akışı Şekil 2.8'de görülmektedir.

Şekil 2.8. Kara kutu giriş çıkış değerleri

Kara kuru testinin adımları:

a. Rastgele giriş değerleri oluşturur.
b. Giriş parametrelerinin en üst ve en alçak değerleri test edilir.
c. Sayısal değer girişi varsa ’0’ mutlaka test edilmelidir.
d. Aşırı yükleme ile değerler test edilir.

Kara kutu testinin avantajları:

a. Büyük ölçekte kodlarda etkilidir.
b. Test edicinin programlama bilgisi olması gerekmez.
c. Belirsiz durumları giderir.

Beyaz kutu:

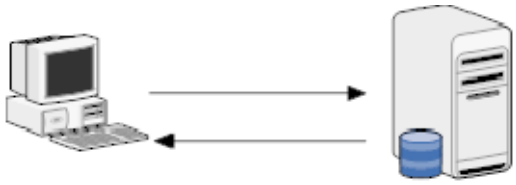
Kara kutu test yaklaşımının aksine kod yapısıyla ilgilenir. Kod akışları ve dizayn kontrolü yapılır. Kaynak koda erişim gerektirir. Yazılım güvenliliği ve açıkları konularında durulur ve gerekli düzeltmeler yapılır.

Beyaz kutu test avantajları:

a. Program hatalarının hızlı bulunabilmesini sağlar.
b. Tasarım özelliklerinin doğrulanmasını sağlar.
c. İstenmeyen özelliklerin koddan çıkarılmasını sağlar.

2.9.3. Performans testi

Performans testi uzaktan veri erişimi olan sistemlerde kullanılır. Yazılımın farklı durum ve konfigurasyonlarda verdiği cevap zamanı olarak ifade edilebilir. Şekil 2.9'da performans test donanımları görülmektedir.



Şekil 2.9. Performans testi donanımları

Performans sorunlarında

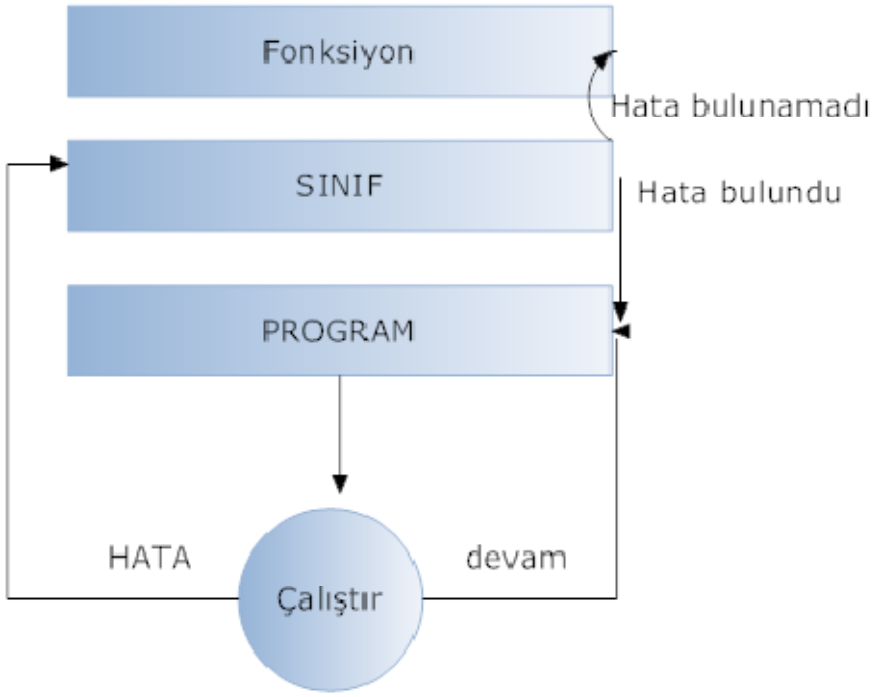
a. Ne kadar uygulama uzaktan çalıştırılıyor?
b. Veri tabanında ne kadar güncelleme yapılacak?
c. Her işlemde ne kadar veri gönderilecek?

Yukarıdaki soruları sorup cevap alınmaya çalışılırken performans karşılaştırması için aşağıdaki ölçütler göz önüne alınmalıdır:

a. İletim hızı
b. Veri transfer oranı
c. Bant genişliği
d. Etkinlik
e. Güvenilirlik

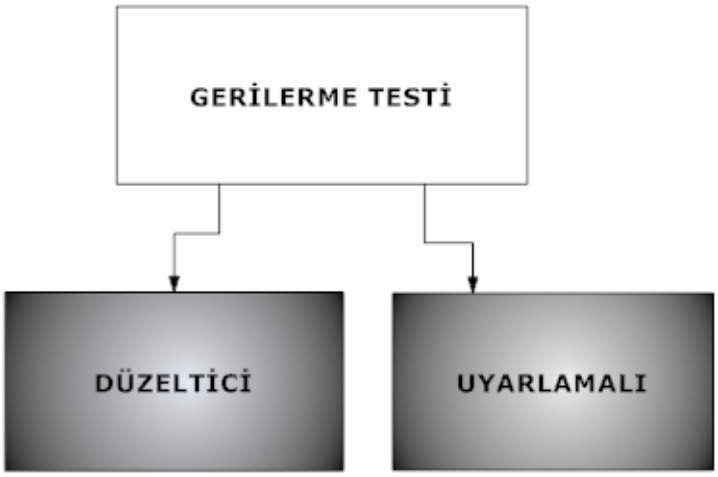
2.9.4. Gerileme testi

Gerileme testi geri dönüş hatalarını ortaya çıkarır. Yani yazılım çalıştığında bir önceki istenen sonucu vermiyorsa gerileme testi uygulanır. En alt adımdan geriye doğru yapılan yazılım aşamaları takip edilip hatanın kaynağı bulunmaya çalışılır. Gerileme test akışı Şekil 2.10'da görülmektedir:



Şekil 2.10. Gerileme test akışı

İki tip gerileme testinden bahsedebiliriz. Bunlar uyarlamalı(adaptive) ve düzeltici(corrective). Uyarlamalı gerilemede test durumları değiştirilir. Tekrar yeni test adımları ile test işlemi yapılarak hatalar bulunmaya çalışılır. Düzeltici yaklaşımda ise test durumu değiştirilmez. Aynı test adımları tekrar edilerek hata bulunmaya çalışılır. Sistem üzerindeki tüm hatalar giderilinceye kadar işlemler tekrar edilir. Sisteme yeni kod parçaları, yeni modül veya yeni bir donanım ve bu donanıma bağlı bir yazılım parçası eklenmişse gerileme testi uygulanmalıdır. Çünkü yeni sistemde daha önce yapılan testlerin geçerliliği kalmaz. Gerileme testi ile ilgili yapılan işlemler Şekil 2.11'de gösterilmiştir.



Şekil 2.11. Gerileme testi seçenekleri

2.9.5 Sistem Testi

Sistem testinde güvenilirlik, güvenlik, sürdürülebilirlik gibi konular test edilir. Yazılımın farklı donanımlarla nasıl çalıştığı üzerinde durulur. Bu noktada yazılım iç yapısı ile ilgilenilmediği için kara kutu yaklaşımında ele alınabilir.

Geliştirilen yazılımın performans, güvenirlik, işlevsellik gibi özelliklerini değerlendiren testlerdir. Birim/sınıf testlerde ve tümleştirme (entegrasyon) testlerinde geliştirilen yazılımın tasarıma uygun olarak geliştirildiği doğrulanır. Sistem testleri ise, müşterinin sistemden istediklerini doğrulamayı amaçlar. Bu nedenle sistem test durumlarında sistem gereksinimleri temel alınarak, sistemin çalışacağı, gerçek ortamda karşılaşılabilecek olan senaryolar tanımlanır [7].

Sistem testleri, gereksinimlerin karşılandığını doğrulamak için entegre edilmiş sistemi tamamen test eder. Bir yandan, sistem performans, güvenlik, dayanıklılık veya harici sistemlerle etkileşim gibi fonksiyonel olmayan gereksinimlere karşı doğrulanabilir. Diğer yandan, sistem tarafından uygulanan işlevsel özellikleriyle karşılaştırılabilir [4].

Birleştirilmiş sistem testi, yazılım test yaşam döngüsünün temel evresidir. Buna ilaveten, hatalar ilk başlarda tespit edilirse test etmek için gerekli çaba azalır.
Birleştirilmiş sistem testi; sağlamlık testi, dokümantasyon testi, performans testi, stres testi ve sınır testini içerir. Sistem testi, yazılım geliştirme yaşam döngüsünün son evreleri esnasında uygulanır. Sistem testinin, hem birleştirilmiş sistem testini hem de kabul testini içerdiği düşünülebilir. Sistem testinin temel amaçlarından biri, yazılım sisteminin belirli bir zaman diliminde davranışlarını gözlemleyerek kullanıcılar için hazır olup olmadığını kararlaştırmaktır. Gereksinimlerin uygunluğu için entegrasyon testini tam bir şekilde test eder. Sistem testi, bir sistemin fonksiyonel, performans, stres ve kaynak gereksinimlerin tümüne uygulanmalıdır [22]. Sistem testleri, kullanıcı kabul testlerinden bir önceki adım olarak yazılım ve donanım entegrasyonundan sonra "sistem test planlama"ya göre gerçekleştirilir. Sistem testlerinin ilk adımı olarak işlevsel gereksinimlerin doğrulanması gerçekleştirilir. Bu adımdan bir sonraki adım ise işlevsel olmayan gereksinimlerin karşılandığını göstermek amacıyla işlevsel olmayan testler gerçekleştirilir. Bu testlerden bazıları şunlardır:

a. Grafik arayüz testi
b. Kullanılabilirlik testi
c. Performans Testi
d. Hata yakalama testi
e. Yükleme Testi
f. Stres Testi
g. Güvenlik Testi
h. Ölçekleme Testi
i. Uyumluluk testi
j. Geri alma Testi
k. Kullanıcı arayüz Testi
l. Yükleme Testi

Grafik arayüz testinde arayüzün kullanım kolaylığı test edilir. Kullanıcı işlemlerine erişim kolay olmalıdır.

Kullanılabilirlik testinde geliştirilen ürünün anlaşılabilirliği test edilir. Kullanıcının yazılımla ilgili yaptığı geri dönüşler dikkate alınır. Kullanılabilirlik testi (Usability testing); Kullanıcı-sistem etkileşimini ve ergonomisini değerlendirmek üzere gerçekleştirilen testlerdir.

Performans testi (Performance testing); Sistem çıktılarının belirlenen ve kabul edilebilecek olan zaman dilimi içerisinde üretebildiğinin değerlendirilmesinin yapılabilmesi için gerçekleştirilen testlerdir [7]. Performans testi, yapılan işlem zamanlarının hesaplanması ve sistemin dar boğazlarının tespit edildiği bölümdür.

Hata yakalama testi, uygulamanın sağlamlığını arttırmak anlamına gelmektedir [25]. Vega Subgraph aracı ile yaptığımız testler aynı zamanda hata yakalama testi olarak nitelendirilebilir.

Yükleme testi, geliştirilen ürünün farklı donanımsal özelliklerdeki sistemler üzerinde denenip karşılaştırılmasıdır. Selenium IDE ile yapılan testler, beklenen sonuçlar ile gerçek sonuçların karşılaştırılıp değerlendirilmesidir.

Stres testi, sistemdeki veri tabanına farklı tipte değişik aralıklarda değerlerin kaydedilip sonuçların gözlemlenmesidir. Stres testi; Sisteme girdi oranı sistem tasarım oranını aştığı zaman sistemin davranışını gözlemlemek üzere gerçekleştirilen testlerdir.

Güvenlik testi, eğer sistemde bir veri tabanı mevcutsa bu veri tabanı açıklarının tespit edilmesi ayrıca sistem web tabanlı çalışıyorsa web güvenlik açıklarının tespit edilmesidir. Güvenlik testi (Security testing); Sistemin izinsiz kullanım teşebbüslerindeki davranışlarının değerlendirilmesi için gerçekleştirilen testlerdir [7]. Sistemin zararlı dış müdahalelerden ve bilgi hırsızlığından korunabildiğinin kanıtlanmasıdır [26].

Ölçekleme testi, sistemden elde edilen değerlerin aralılarının belirlenmesi ve gruplandırılması işlemidir. Gerçek zamanlı sistemlerde yazılım işlem süresinin

bilgisayara dayalı sistem ile uyarlılığının sınanmasıdır. Performans sınanması, her test basamağında uygulanmaktadır. Ancak, sistemin performansı tam olarak sistemin bütünleştirilmesinden sonra anlaşılabilmektedir [26].

Konfigürasyon ve uyumluluk testleri (Configuration and compatibility testing); Geliştirilen sistemin farklı platformlarda ve donanımlarda nasıl davrandığının değerlendirilmesi için gerçekleştirilen testlerdir.

Geri alma testi (Recovery testing); Bir hata durumunda sistemin otomatik veya elle yeniden normal duruma dönmesini değerlendirmek için gerçekleştirilen testlerdir.

Kullanıcı arayüzü testi (User interface testing); Kullanıcının ve yazılımın grafik gösterimi olarak nasıl bir etkileşim içerisinde olacağını, kullanıcının klavye, ekran veya fare ile sisteme vereceği girdilerin sistem tarafından nasıl işleneceğini değerlendirmek için gerçekleştirilen testlerdir. Sistem testleri sırasında ortaya çıkan hatalar proje hata yönetim sürecine göre raporlanır ve gerekli düzeltme işlemleri gerçekleştirilir. Gerekli düzeltmelerden sonra, düzeltmelerden sistemin kalanının etkilenmediğinin değerlendirilmesi için sistem üzerinde yineleme testleri gerçekleştirilir [7].

Yükleme testleri (Load testing); Aynı anda maksimum sayıda sanal kullanıcı tarafından yapılan testlerdir.

BÖLÜM 3. UZAKTAN EĞİTİME GİRİŞ

3.1. Uzaktan Eğitim Tanımı

Web tabanlı uzaktan eğitim, öğrencilerin fiziksel olarak bir yerde bulunmasını zorunlu hale getirmeksizin, günümüzdeki teknolojinin imkanlarından yararlanılarak, öğrenci ve öğretmenlerin bir sanal dershane ortamı içerisinde değişik şekillerde karşılıklı gelmesi durumudur. Bilinen geleneksel öğrenme-öğretme yöntemlerindeki sınırlılıklar nedeniyle sınıf içi etkinliklerin yürütülme olanağı bulunmadığı durumlarda eğitim çalışmalarını planlayanlar ve uygulayanlar ile öğrenenler arasında iletişim ve etkileşimin özel olarak hazırlanmış öğretim üniteleri ve çeşitli ortamlar yoluyla belli bir merkezden sağlanma durumudur. Öğreten ile öğrencinin birbirinden uzak mesafede olmalarına karsın aynı veya ayrı zamanlı olarak bir araç vasıtasıyla iletişim kurmaları mümkündür [27].

Uzun yıllardan beri dünyada ve ülkemizde açık öğretim veya yaygın öğretim adı altında ön lisans ve lisans düzeyinde örgün eğitimin haricinde eğitim verilmektedir. İletişim teknolojisinin gelişmesiyle birlikte yaygın öğretime olan talep de artmıştır. Günümüzde uzaktan eğitimde kullanılan araçların çeşitlenmesi ve güçlenmesi nedeniyle bu öğretim türüne olan ilgi daha da artacaktır [4].

3.1.1 Uzaktan eğitimin tarihçesi

Uzaktan öğrenimin ilk başlangıcı olarak kabul gören mektupla öğrenimdi. Herhangi bir okul veya yetki verilmiş kurumlar eğitimlerini posta yolu ile başladılar ve bu öğretim yolu zamanla gelişme gösteren bir yöntem olarak kabul gördü. Bu yöntem mesleki eğitim alanında o kadar gelişti ki tüm meslek guruplarını kapsayacak şekilde büyüdü. Bu sayede evinden çıkamayacak durumda olan engelli insanlar mektupla eğitime, kurslarına akın akın katılımda bulundular. Bu istek ve talebin ilk başlarda kaygıyla bakılan bir sistemin giderek büyümesine sebep oldu. Özellikle fiziksel

engelliler ve eve bağlı olanlar için ideal olan bu mektupla öğrenim kursları, körler ve sağır çocukların anne-babaları için özel programlar da düzenlemekteydi. Günümüzde mektupla öğrenimi is çevreleri, sivil toplum kuruluşları, dernekler ve silahlı kuvvetler yoğun biçimde yararlanan kurumlardır. Uzaktan eğitimin tarihini su şekilde açıklamak en güzeli olacaktır [28]. "Bilgisayar ekranında gerçekleştirilen uzaktan eğitim programı tas ve tuğladan yapılan binalarda diğer bir ifadeyle kampüs içinde gerçekleştirilen eğitimin yerini mi alacak ya da geleneksel eğitim sistemi ile birlikte mi yürütülecek? Bu ve bunun gibi sorular gerek ABD gerekse dünya üzerinde birçok ülkenin birlikte tartıştığı konular arasında yer almaktadır. Eğitim konusunda bası çeken çevrelerde uzaktan eğitim programlarının olumlu bir sonuç mu doğuracağı yoksa yıkıcı bir etki mi oluşturacağı hala zihinleri meşgul eden bir soru olarak durmaktadır." "Uzaktan eğitimin başlangıcı sayılabilecek mektupla öğrenim, bir okul veya yetkili kurum tarafından posta vasıtasıyla yürütülen öğretim yöntemidir. İlk olarak 1728'de Boston gazetesi mektup ile stenografi (o ortamda kişilerin söyledikleri sözleri özel işaret ve hareketler hızlı yazmaya yarayan bir yazı çeşidi) dersleri verildi. Bu 20 Mart 1728 tarihinde Caleb Phillipps tarafından Boston Gazetesine verilen bir ilan ile duyuruldu. 19'uncu yüzyılın ortalarında İngiltere, Fransa, ABD ve Almanya'da hızla yayıldı. 1840'ta İngiliz eğitimci Sir Isaac Pitman postayla (Penny Post`u kullanarak) stenografi öğretmiştir. 1856`da Fransız Charles Toussaint ve Alman Gustav Langenscheidt Berlin`de mektup ile eğitim okulu kurmuştur. Mektupla eğitim üniversitesi, gelişimini ve yaygınlaşmasını, İngiltere'deki Cambridge Üniversitesi'nden İskoç eğitimci James Stuart tarafından verilen kampüs dışı derslere borçludur. 1870'lerde Illinois Wesleyan Üniversitesi evde öğrenim programı başlattı. 1873`te Boston`da bulunan toplumu evde çalısmaya tesvik etme (Society to Uncourage at Home) isimli eğitim kurumu Anna Eliot Ticknor tarafından kuruldu ve ölümü olan 1897`ye kadar kendisi çalıştı. Bu kurumun öğrencilerinin büyük kısmı kadınlardan oluşmuştur.1883'te New York - Ithaca'da bir "Mektupla Öğretim Üniversitesi" kuruldu. 1882'de William Rainey Harper Chautauqua, New York'ta bir mektupla öğrenim programı geliştirdi ve yeni kurulan Chicago Üniversitesi'nin ilk başkanı olduğunda (1891) bu yönteme devam etti. 1880'lerde Thomas J. Foster'in başlattığı evde - öğrenim kursları 1890'da Uluslar arası Mektupla Öğrenim Okulları halini aldı. 1890'da Avusturalya Queensland

Üniversitesi kampüs dışına açık bir eğitim programı yürütmüştür. 1920'lerde aynı tür bir eğitim metodu Colombia Üniversitesi tarafından gerçekleştirilmiştir.

1914' de bir yasa ise ABD' de mektupla öğrenimin temelleri atılmıştır. ABD'de çok sayıda mektupla öğrenim kurumu mevcuttur; bunların çoğu Ulusal Evde Öğrenim Konseyi'nin onaylı üyesidir. 1939 yılında televizyon aracılığıyla 500`den fazla program televizyon aracılığı ile sunulmuştur. 1946 yılının Güney Afrika Üniversitesi (UNISA) de uzaktan eğitim veren 11 üniversiteden en büyüklerinden biri olan Division of External Study isimli bir bölüm oluşmuştur. 1950 yılında e ABD'de uzaktan eğitim veren askeri amaç düşüncesine sahip uzaktan eğitim uygulamaları yapan sistemler oluşturulmuştur."

3.1.2. Web tabanlı eğitimin avantajları

Eğitim alanındaki yapılan çalışmalar sonucunda bir genelleme yapılacak olursa, öğrencilerin yaklaşık üçte birinin görerek, diğer üçte birinin yaparak ve geriye kalan üçte birinin ise dinleyerek öğrenme yeteneğine sahip olduğu belirlenmiştir. Bu açıdan ele alındığında akla gelen en etkili eğitim, Internet yoluyla yapılan eğitimdir [29]. Web tabanlı eğitim sayesinde, farklı toplum ve gruplar arasında bir denge sağlanarak, fırsat eşitsizliği en aza indirgenmektedir. Internet tabanlı eğitimin olumlu yönlerini ve avantajları aşağıdaki gibi sırlanabilir [30, 31]:

a. Basım ve kırtasiye giderleri gibi birçok giderler en az seviyede tutulmaktadır.

b. Metin tipinde bir sunumdan öte, ses, renk, grafik, animasyon gibi unsurlarla beraber web daha etkili olmaktadır.

c. Eğitim, zamandan ve mekandan bağımsız bir şekilde yürütüldüğünden, sınırsız ve süresiz bir eğitim imkanı ortaya çıkmaktadır.

d. Öğrencilerin, kendi kendilerine bireysel öğrenme yetenekleri gelişmektedir.

e. Bilgilerin kolaylıkla değiştirilebilmesinden dolayı sürekli güncel bilgiler sunulmaktadır.

f. Bilgiye, kaynağından ulaşma imkanı sunulmaktadır.

g. Eğitim, bilgi teknolojilerine dayalı olarak sürdürülmektedir.

h. Gruplar arasında (öğrenci-öğretmen ve öğrenci-öğrenci) çok yönlü bir haberleşme sağlanmaktadır.

i. Geleneksel sınıf ortamında soru sormaktan çekinen veya grup çalışmalarına katılamayan öğrenciler, elektronik ortamda özgüven kazanmaktadır.

j. Sunum, ortamdan, öğrenciden, eğitmenden ve diğer çevre koşullarından bağımsız olduğundan dolayı, öğretimsel tutarlık göstermektedir.

k. Bireysel katılım ve karşılıklı etkileşim gerçekleştiğinden, ilginin artması sağlanmaktadır.

l. İletişim ve ulaştırma gibi alanlarda görülen altyapısal farklılıkların yanında, kültürel ve toplumsal seviye farklılıkların etkili olmamasından dolayı eğitimi demokratikleşmektedir.

m. Seyahat, barınma masrafları ve kişilerin seyahat süresince oluşan üretim kaybının ortadan kalkması ve bu nedenle de birey açısından öğrenim maliyetinin düşmesi sağlanmaktadır.

n. Öğrencilerin, Internet, bilgisayar ve bilgisayar teknolojilerini kullanım becerilerini arttırarak, insan hayatını birebir etkileyen bu öğelerin kullanımına yönelik hazırlık sağlaması gibi sıralanabilir.

BÖLÜM 4. ÖLÇME VE DEĞERLENDİRME SİSTEMİ TASARIMI

4.1. Genel Bilgiler

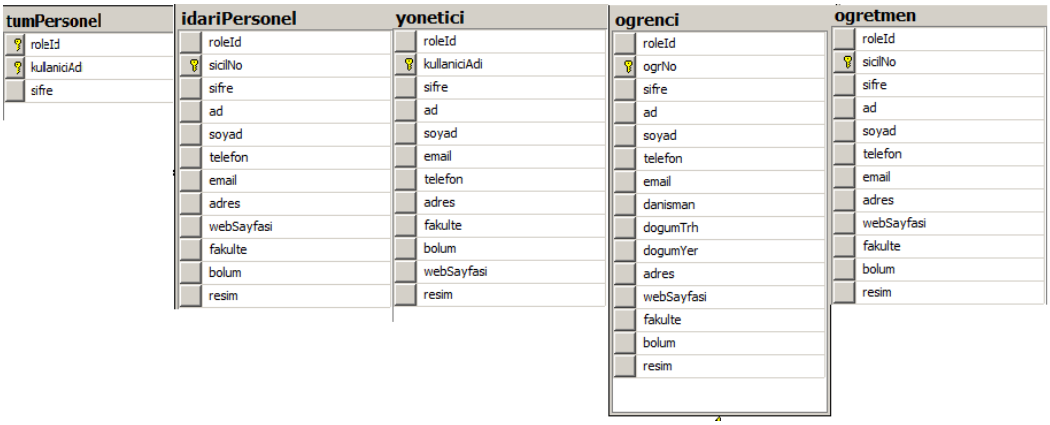
ÖDS Asp.net Framework 4.0 ile kodlanmıştır. Sistem web tabanlı çalışmaktadır. Veri tabanı kısmında SQL Server 2005 kullanılmıştır.

Ölçme değerlendirme sayfası giriş sayfası aşağıda gösterilmiştir. Bu sayfadan kullanıcı adı ve şifre bilgileri ile sisteme giriş yapılmaktadır. Öğrenciler için öğrenci no, idari personel ve öğretmenler için sicil no ve yöneticiler için kullanıcı adı bilgileri giriş için kullanılmaktadır. Tüm personelin yer aldığı tabloda kaydın varlığı sorgulanmakta eğer kayıt varsa rol numarasına göre ilgili tablodan veriler çekilmektedir. "Session" yapıları ile oturum bilgileri tutulmaktadır.

4.2. Ölçme Değerlendirme Sistemi Veri Tabanı Tasarımı

Uzaktan eğitimde web tabanlı bir ÖDS yapılabilmesi için bu sisteme ait bir veri tabanı olması gerekmektedir. ÖDS'ye ait sınav bilgileri ders-sınav, ders-öğretmen veri alışverişi veri tabanı üzerinden yapılmalıdır. Sisteme ait tüm rollere ait bilgiler veri tabanından çekilmeli elde elden sonuçlar veri tabanına kaydedilmelidir.

Uzaktan eğitim ÖDS kullanıcı yetkilendirmesi tabanlı çalışan bir sistem olduğu için "tumPersonel", "ogretmen", "ogrenci" ve "idariPersonel" tabloları oluşturulmuştur. Yeni kaydedilen personel hem "tumpersonel" tablosuna hem de hangi role sahipse rolüne uygun tabloya kaydedilmektedir. Tüm rollere ait kişisel bilgiler veri tabanında tutulmaktadır. Ayrıca sistem üzerinden kullanıcı bilgileri de güncellenebilmektedir. Şekil 3.1'de giriş rollerine ait tablolar gösterilmiştir.

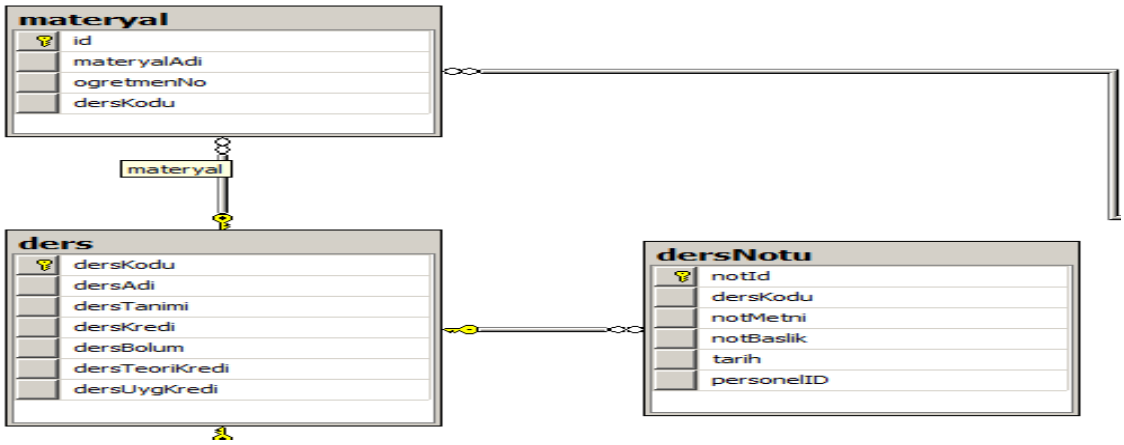


Şekil 3.1. Rol tabloları

Her rol için "roleId" tanımlanmıştır. "roleId" "roles" tablosundan alınmaktadır. Giriş yapılırken "tumPersonel" tablosuna bakılmaktadır. Eğer kullanıcı mevcutsa "roleId" değerine göre uygun tablodan veriler çekilmektedir.

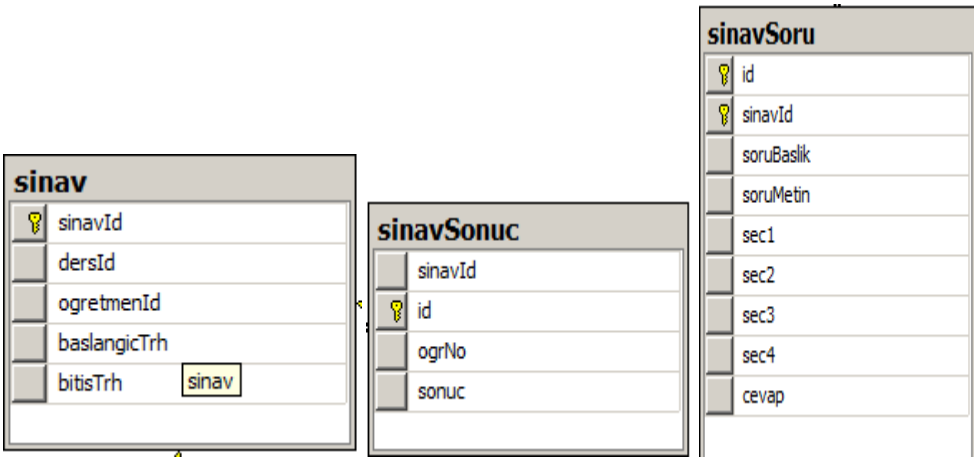
İdari personeller sicil numaralarına göre kaydedilmektedir. Yönetici, öğrenci ve öğretmen girişleri için kullanıcı adı bilgisi, sicil numarası ve öğrenci numarası ile şifre bilgileri kullanılmıştır. Kullanıcılar sistem üzerinden giriş yaparken kullanıcı türü bilgisi sorulma gereği de "tumPersonel" tablosu sayesinde ortadan kalkmıştır. "roleId", "sicilNo", "kullaniciAdi" ve "ogrNo" alanları "primar key" tanımlanmıştır. "roles" tablosu "roleId" ile kullanıcı tablolarındaki "roleId" alanları arasında "Foreign Key" tanımlanmıştır.

Ders bilgileri için kullanılan tablolar Şekil 3.2'de gösterilmiştir.



Şekil 3.2. Materyal akış diyagramı

Her dersin ders kod bilgileri “primary key” olarak tutulmaktadır. Derse ait özellik bilgileri tanımlanmış ve materyal tablosu ile ilişkilendirilmiştir. Materyal tablosu ile ders tablolarındaki “dersKodu” alanları “Foreign Key” olarak tanımlanmıştır. Ders tablosunda bulunmayan bir ders koduna ait materyal girilememektedir.

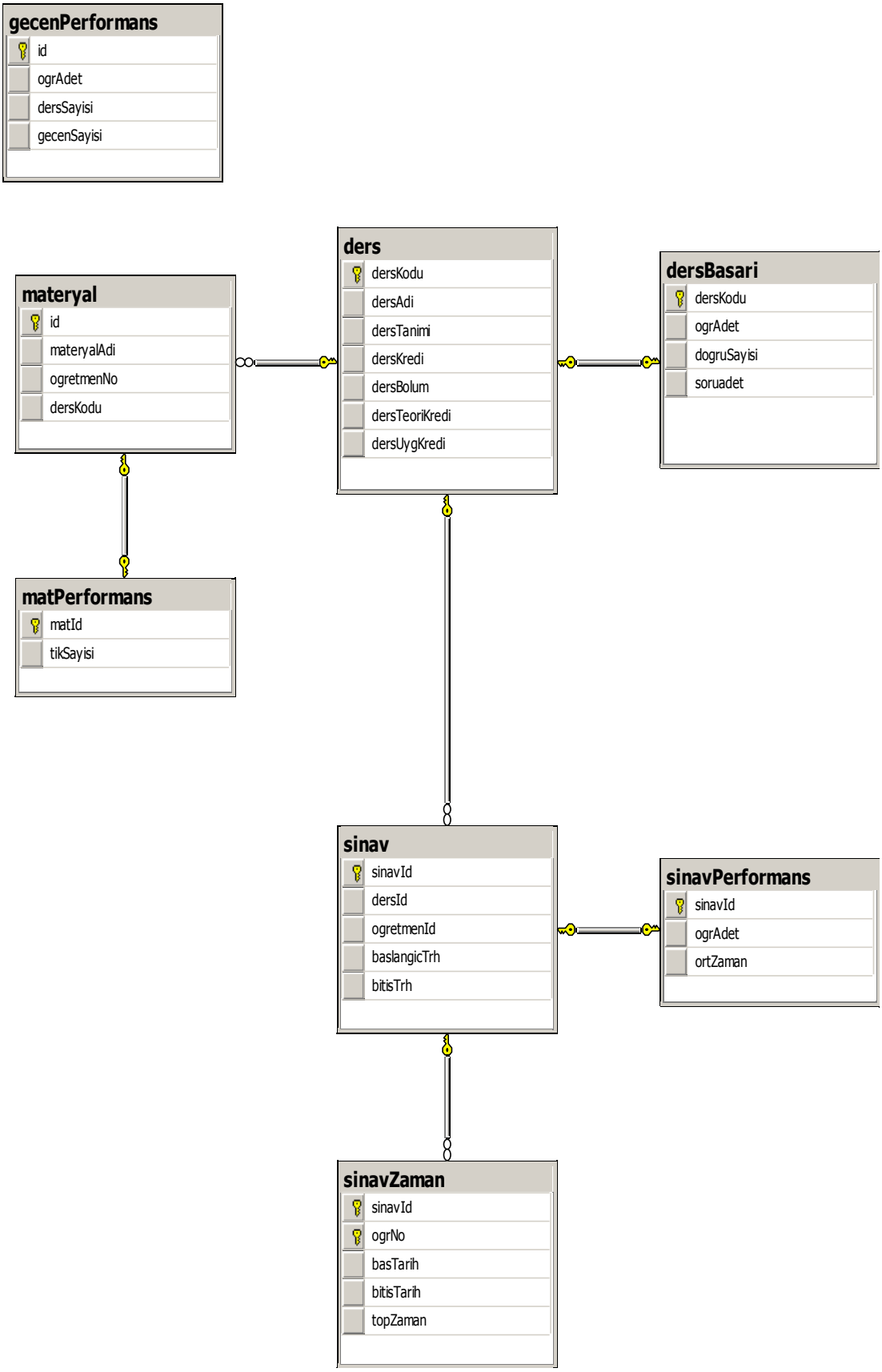


Şekil 3.3. Sınav tablo görünümleri

Sınav için kullanılan tablolar Şekil 3.3’de gösterilmektedir. Sınav bilgileri “sinavId” ile tutulmakta “dersId” alanı ile hangi derse ait olduğu tutulmaktadır. Sınav sonuçları sınav sonuç tablosunda tutulmaktadır. “sinav” tablosu ile “sinavSonuc” tablosu arasında “sinavId” alanları “Foreign Key” olarak tanımlanmıştır.

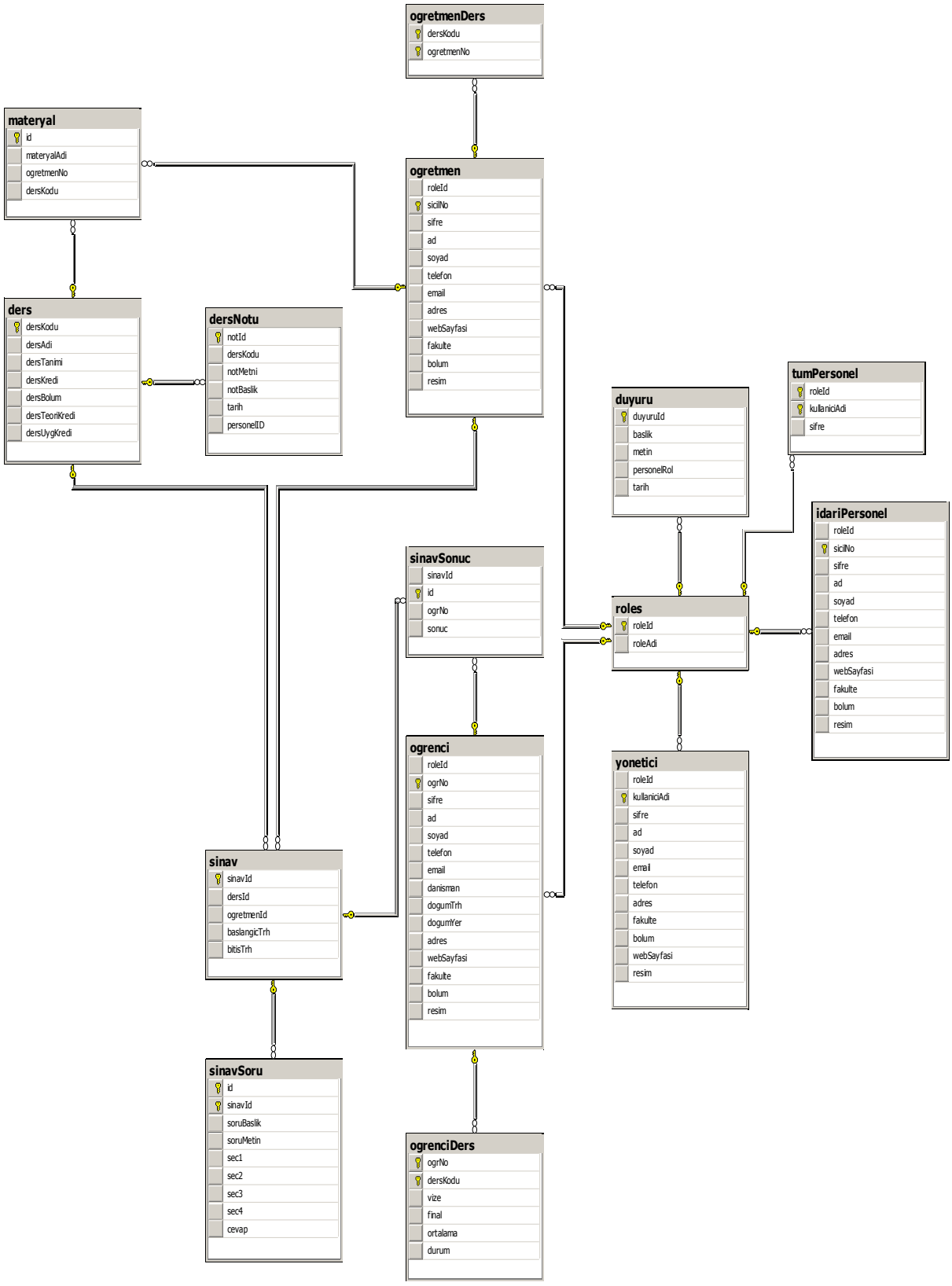
Sınav açma ve kapama işlemleri öğretmen rolü tarafından yapılmaktadır. Veri tabanında sınav bilgileri sınav başlama ve bitiş tarihi süresince tutulmakta ve sınav bitiş tarihinden sonra veri tabanından silinmektedir. “sinav” tablosunda sınavı açan öğretmen “ogretmenId” ile tutulmakta, bu sayede sınav sonuçları, sınav soru görüntüleme işlemleri, soru ekleme işlemleri sadece sınavı açan personel tarafından yapılabilmekte ve sınavı açan öğretmenden dersi alan öğrenciler sınav girişi yapabilmektedirler.

Yazılım test teknikleri ile elde edilecek sonuçlar için kullanılan tablo ilişki görünümü Şekil 3.4’deki gibidir:



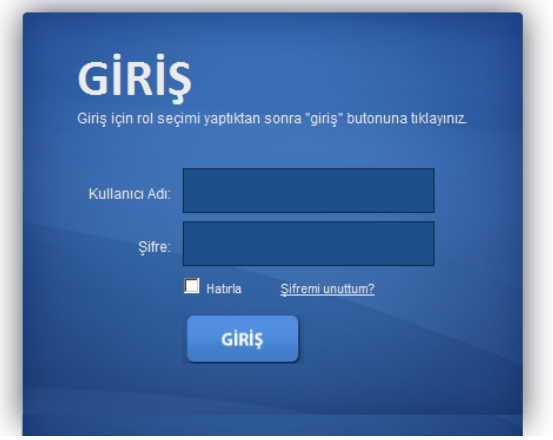
Şekil 3.4. Sınav ve ders tablo diyagramı

Personel ve öğrenci ilişkilerini gösteren diyagram Şekil 3.5'deki gibidir.



Şekil 3.5. Genel işlem tabloları diyagramı

4.3. Giriş Sayfası



Şekil 4.1. Ölçme sistemi giriş sayfası

Giriş için kullanılan kodun bir bölümü aşağıda gösterilmektedir.

```
string sql = "Select * From tumPersonel Where    kullaniciAdi=@User and sifre=@pass";
            SqlConnection bag = new SqlConnection(constr);
            bag.Open();
            SqlCommand cmd = new SqlCommand(sql, bag);

            cmd.Parameters.AddWithValue("@User", kullanici.Value);
            cmd.Parameters.AddWithValue("@pass", sifre.Value);

            SqlDataReader dr = cmd.ExecuteReader();

            if (dr.Read())
            {
               rol = Convert.ToInt32(dr[0].ToString());

               switch(rol)
               {
                  case 1:
```

```
                string sql2 = "Select * From yonetici Where kullaniciAdi=@User and
sifre=@pass";
                SqlConnection bag2 = new SqlConnection(constr);
                bag2.Open();
                SqlCommand cmd2 = new SqlCommand(sql2, bag2);

                cmd2.Parameters.AddWithValue("@User", kullanici.Value);
                cmd2.Parameters.AddWithValue("@pass", sifre.Value);

                SqlDataReader dr2 = cmd2.ExecuteReader();
                dr2.Read();
                Session["kullaniciTipi"] = "Yönetici";
                Session["KullaniciAdi"] = Convert.ToString(dr2[1]);
                Session["Adi"] = Convert.ToString(dr2[3]);
                Session["Soyad"] = Convert.ToString(dr2[4]);
                Session["Resim"] = dr2["resim"].ToString();
                Response.Redirect("~/giris/Default.aspx");
            break;
              case 2:
                string sql3 = "Select * From ogretmen Where  sicilNo=@User and sifre=@pass";
                SqlConnection bag3 = new SqlConnection(constr);
                bag3.Open();
                SqlCommand cmd3 = new SqlCommand(sql3, bag3);

                cmd3.Parameters.AddWithValue("@User", kullanici.Value);
                cmd3.Parameters.AddWithValue("@pass", sifre.Value);

                SqlDataReader dr3 = cmd3.ExecuteReader();
                dr3.Read();
                Session["kullaniciTipi"] = "Öğretmen";
                Session["KullaniciAdi"] = Convert.ToString(dr3[1]);
                Session["Adi"] = Convert.ToString(dr3[3]);
                Session["Soyad"] = Convert.ToString(dr3[4]);
                Session["Resim"] = dr3["resim"].ToString();
                Response.Redirect("~/giris/Default.aspx");
                break;
              case 3:
                string sql4 = "Select * From ogrenci Where  ogrNo=@User and sifre=@pass";
                SqlConnection bag4 = new SqlConnection(constr);
```

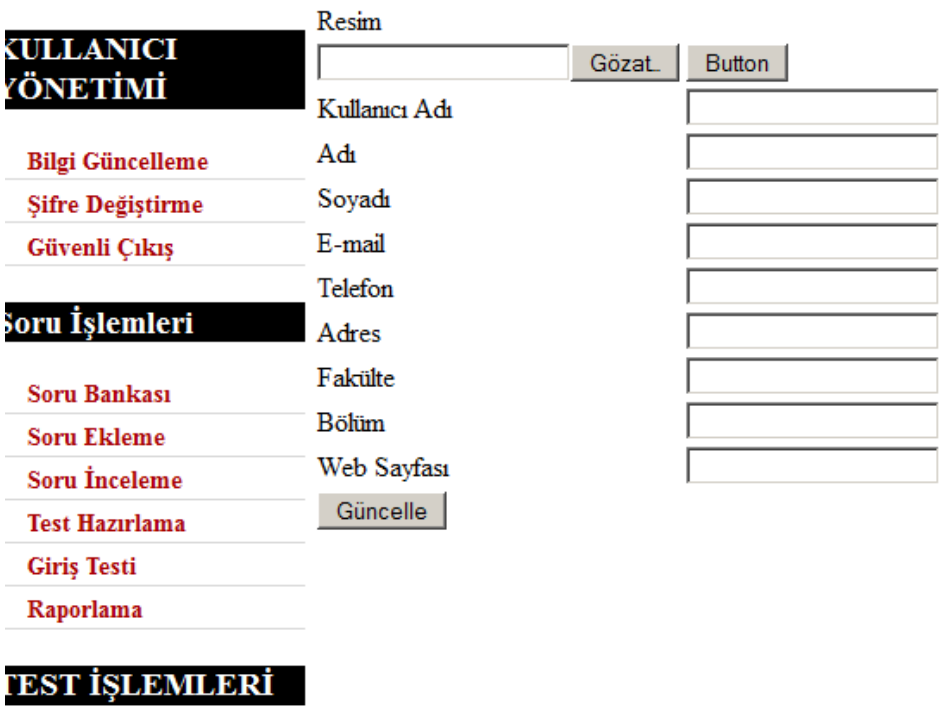
```
            bag4.Open();
            SqlCommand cmd4 = new SqlCommand(sql4, bag4);

            cmd4.Parameters.AddWithValue("@User", kullanici.Value);
            cmd4.Parameters.AddWithValue("@pass", sifre.Value);

            SqlDataReader dr4 = cmd4.ExecuteReader();
            dr4.Read();
            Session["kullaniciTipi"] = "Öğrenci";
            Session["KullaniciAdi"] = Convert.ToString(dr4[1]);
            Session["Adi"] = Convert.ToString(dr4[3]);
            Session["Soyad"] = Convert.ToString(dr4[4]);
            Session["Resim"] = dr4["resim"].ToString();
            Response.Redirect("~/giris/Default.aspx");
            break;
        case 4:
            string sql5 = "Select * From idariPersonel Where  sicilNo=@User and sifre=@pass";
            SqlConnection bag5 = new SqlConnection(constr);
            bag5.Open();
            SqlCommand cmd5 = new SqlCommand(sql5, bag5);

            cmd5.Parameters.AddWithValue("@User", kullanici.Value);
            cmd5.Parameters.AddWithValue("@pass", sifre.Value);

            SqlDataReader dr5 = cmd5.ExecuteReader();
            dr5.Read();
            Session["kullaniciTipi"] = "İdari Personel";
            Session["KullaniciAdi"] = Convert.ToString(dr5[1]);
            Session["Adi"] = Convert.ToString(dr5[3]);
            Session["Soyad"] = Convert.ToString(dr5[4]);
            Session["Resim"] = dr5["resim"].ToString();
            Response.Redirect("~/giris/Default.aspx");
            break;
```

Giriş yapıldıktan sonra Şekil 4.2'deki sayfa görüntülenmektedir. Yönetici bu sayfa üzerinden bilgilerini güncelleyebilmektedir. Yönetici rolündeki kişiler aynı zamanda öğretmenlerin tüm işlemlerini de yapabilmektedirler.

4.4. İşlemler Sayfası

Şekil 4.2. Yönetici giriş sayfası

Bilgi güncellemeleri için Şekil 4.3'teki sayfa kullanılmaktadır. Güvenli çıkış butonuna tıklandığında oturum bilgisi silinerek sonlandırılmaktadır. Bilgi güncelleme sayfasından kişisel bilgiler ve resim güncelleme işlemleri yapılabilmektedir. Kişisel resimlerin yol bilgileri veri tabanında tutulmakta ve resimler proje klasörüne kopyalanmaktadır. Şekil 4.3'de Kullanıcı bilgileri güncelleme ekranı görülmektedir.



Şekil 4.3. Kullanıcı bilgileri güncelleme ekranı

4.5. Soru Bankası Sayfası

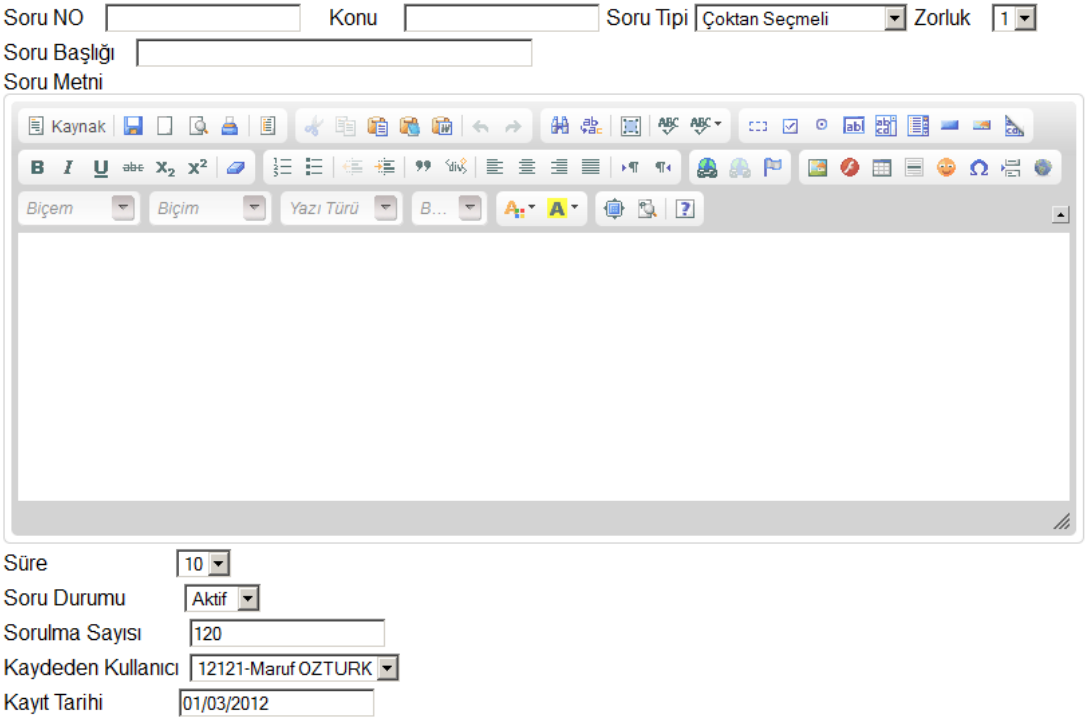
Test sonuçları, test değer kümesini oluşturmaktadır [32]. Soru bankası sayfasından da test değer kümesi elde edilmiştir. Soru bankası bölümünden eklenen sorular görülmektedir. Data Grid ile eklenen sorular veri tabanından çekilmektedir. Şekil 4.4'te soru bilgilerinin listelendiği soru bankası sayfasından bir bölüm görülmektedir.

id	sinavId	soruBaslik	soruMetin
1	1	Best cpu scheduling?	choose the correct answer
111	2	ip katmanı	katmnalardan ip olanı seciniz
123	1	en iyi sırlamayı seçiniz	en iyi sıralama seçeneklerden seçiniz
444	1	zxc	zxczxc
4555	1	asdasdas	zxczxczxczxc
543534	1	asdasd	zxczczx
555	1	pi sayısı	pi 14
555	2	pi sayısı	pi 14
56756	1	zxczxc	asdasdsa <input name="yeni" type="button" value="zxczxczxc" />
6	1	test	zczxczxczxcz

Şekil 4.4. Soru bankası ekranı

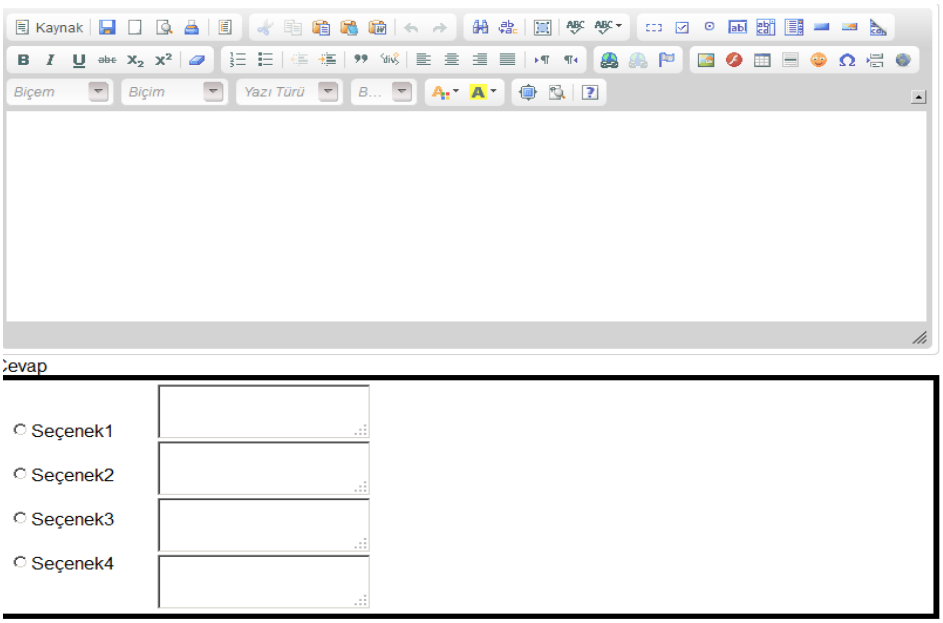
4.6. Soru Ekleme Sayfası

Soru ekleme ekranından sorunun tipine soru ekleme işlemi yapılmaktadır. Şekil 4.5'te soru ekleme ana ekranı gösterilmiştir. Soru Tipi bölümünden sorunun tipi seçilmektedir. Sorunun zorluk derecesi seçilebilmektedir. Sorulara isteğe göre soru numarası ataması yapılabilmektedir. Soru başlığı atamasından sonra CK editör ile soru metni hazırlanmaktadır. Soru ekleme sayfası Şekil 4.5'te görülmektedir.



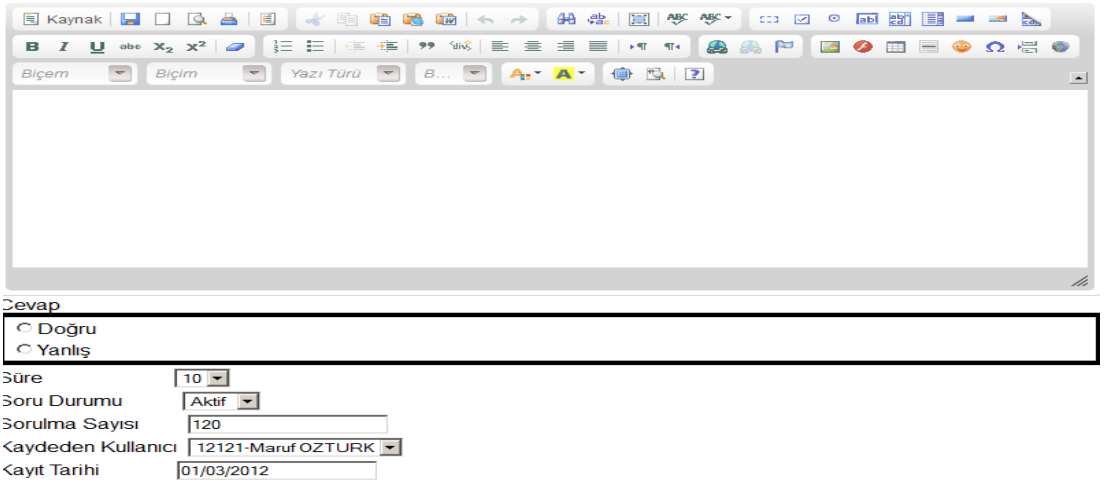
Şekil 4.5. Soru ekleme sayfası

Çoktan seçmeli seçeneği seçildiğinde aşağıdaki gibi çoktan seçmeli seçenekler için bir ekran gelmektedir. Buradan seçenek içerikleri oluşturulmaktadır. "Çoktan Seçmeli" seçeneği seçildiğinde dört adet seçenek içeren panel CKEditor altında görüntülenmektedir. "Doğru/Yanlış" seçeneği seçildiğinde çoktan seçmeli paneli gizlenmekte ve doğru yanlış seçeneklerini içeren panel açılmaktadır. Boşluk doldurma ve eşleştirme seçenekleri için de ilgili paneller seçilen seçeneğe bağşı olarak görüntülenmektedir.



Şekil 4.6. Seçenek ekranı

Doğru/yanlış seçeneği seçildiğinde soru metin bilgileri girilmekte ve eğer metin doğru ise doğru, metin yanlışsa yanlış seçeneği işaretlenip soru kayıt işlemi yapılmaktadır. CK editör soru metninin hazırlanmasını hem yöneticiler hem de öğretmenler için kolaylaştırmaktadır. Ayrıca sorular için soru bitiş süreleri sorunun yayınlanıp yayınlanmamasına ilişkin aktif/pasif seçeneği, sorunun sorulma sayısı gibi bilgiler, soruyu kimin eklediğine ve hangi tarihte eklendiğine dair de bilgiler yer almaktadır. Şekil 4.7'de doğru/yanlış soru tipine ait ekran görülmektedir.



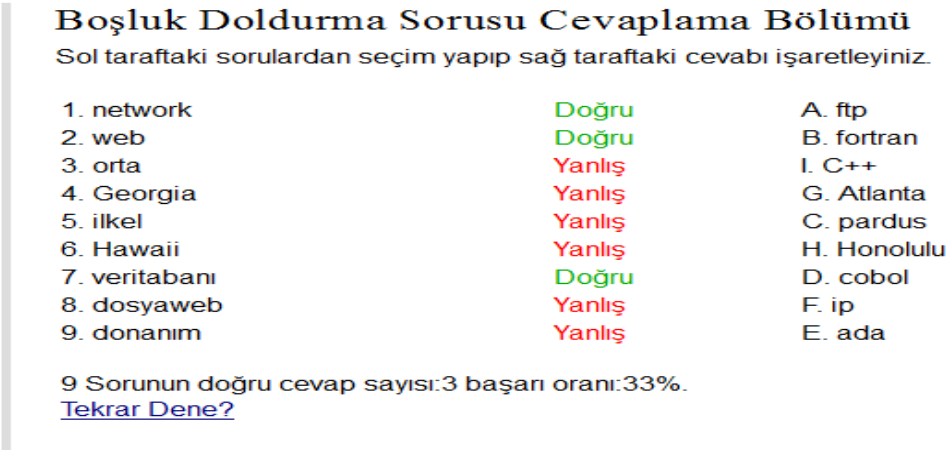
Şekil 4.7. Doğru/Yanlış soru tipi ekranı

4.7. Eşleştirme Cevap Sayfası

Cevapların kontrol edildiği bölüm Şekil 4.8'de gösterilmiştir.

Şekil 4.8. Eşleştirme soru sayfası

Cevap verilirken eğer eşleştirme seçeneği seçili ile soru hazırlandıysa sol ve sağ taraftan bir çenek seçilip diğeri işaretlenebilmektedir. "Cevapları kontrol et" butonuna tıklandığında sonuçlar kontrol edilmekte ve başarı oranı ve doğru cevap sayısı gösterilmektedir. Şekil 4.9'da eşleştirme sınavı değerlendirme sonuçları görülmektedir.

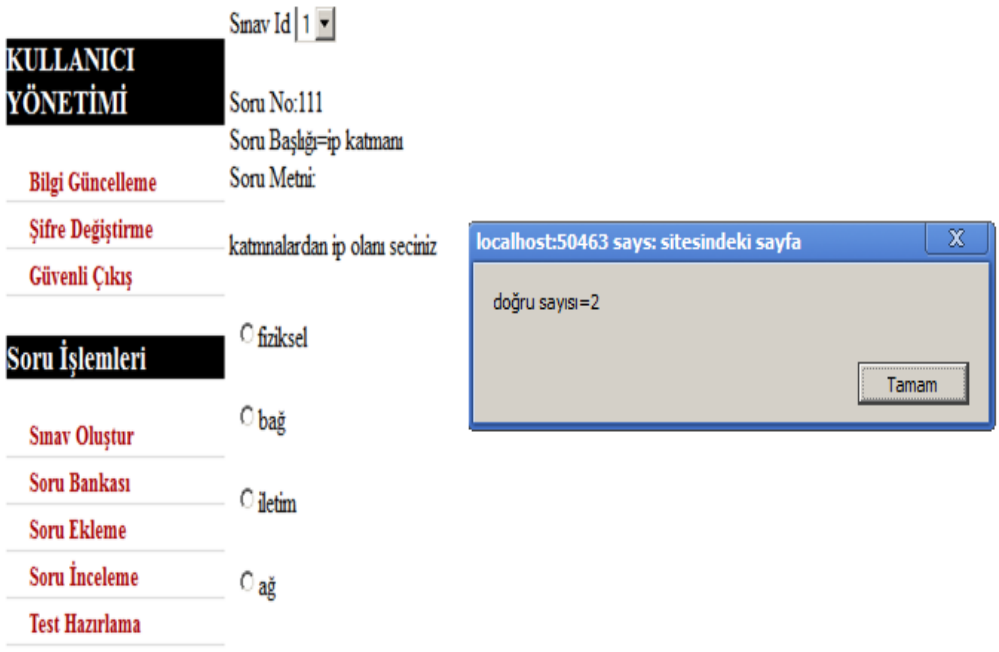


Şekil 4.9. Eşleştirme sınav sonuç ekranı

Sorular ve cevaplar veri tabanından çekilmektedir. Soruların ve cevapların sıralaması bir javascript fonksiyonu sayesinde rastgele değiştirilmektedir. Böylece cevapların sorular ile aynı sırada olması engellenmiş olur.

4.8. Çoktan Seçmeli Soru Cevaplama Sayfası

Çoktan seçmeli soru sayfası Şekil 4.9'da görülmektedir. Seçilen sınav numarası ile ilgili sorular listelenmektedir. Sorular "coktanSecmeli.aspx" dinamik olarak sayfaya eklenmektedir. Soru tipi "Çoktan Seçmeli" olan soru adedine göre sorular listelenmektedir. Bunun için "Literal" tipi değişkenler kullanılmıştır. Soru numaraları eklenme sırasına göre verilmektedir. "Cevapları Gönder" butonu ile doğru sayısı hesaplanıp ekranda görüntülenmektedir.



Şekil 4.10. Çoktan seçmeli soru ekranı

Cevaplanan doğru sayısı javascript uyarı çerçevesi ile görüntülenmekte ve sınav bilgileri kayıt altına alınmaktadır. "Doğru/Cevap" ve "KısaCevap" sayfaları da çoktan seçmeli soru cevaplama sayfası gibi soru bilgilerini ve soru şıklarını veri tabanından çekmektedir.

BÖLÜM 5. ÖLÇME VE DEĞERLENDİRME SİSTEMİ TESTLERİ

5.1. Giriş

Önceki bölümlerde yazılım testi hakkında bilgiler verilmiş, yazılım test teknikleri ayrıntılı olarak anlatılmış ve hangi test metotlarının kullanılacağı belirtilmiştir. ÖDS Visual Studio 2010 ile C# programlama dili kullanılarak yazılmış ve veri depolama için SQL Server 2005 veri tabanı kullanılmıştır. Kullandığımız yazılım geliştirme araçları ile seçeceğimiz test araçlarının uyumlu çalışması gerekmektedir. Bu bölümde web sayfalarının güvenlik testi için Vega Subraph, veri tabanı testi için SQLMon ile SqlQueryStress ve Visual Studio 2010 PerformanceAnalyzer araçları ile yapılan testler ayrıntılı olarak anlatılmıştır.

5.2. Selenium IDE ile Yapılan Testler

5.2.1. Giriş sayfasının test edilmesi

Selenium IDE ile yaptığımız işlemler dinamik test sınıfına girmektedir. Çünkü sisteme belli giriş değerleri verilip çıkış değerleri gözlemlenmekte aynı zamanda sistem fonksiyonlarının doğrulukları onaylanmaktadır. Yapılan geliştirmeler platformun geçerlenmesi testleridir. Giriş sayfasının test edilmesi için Liste1'deki kod oluşturulmuştur.

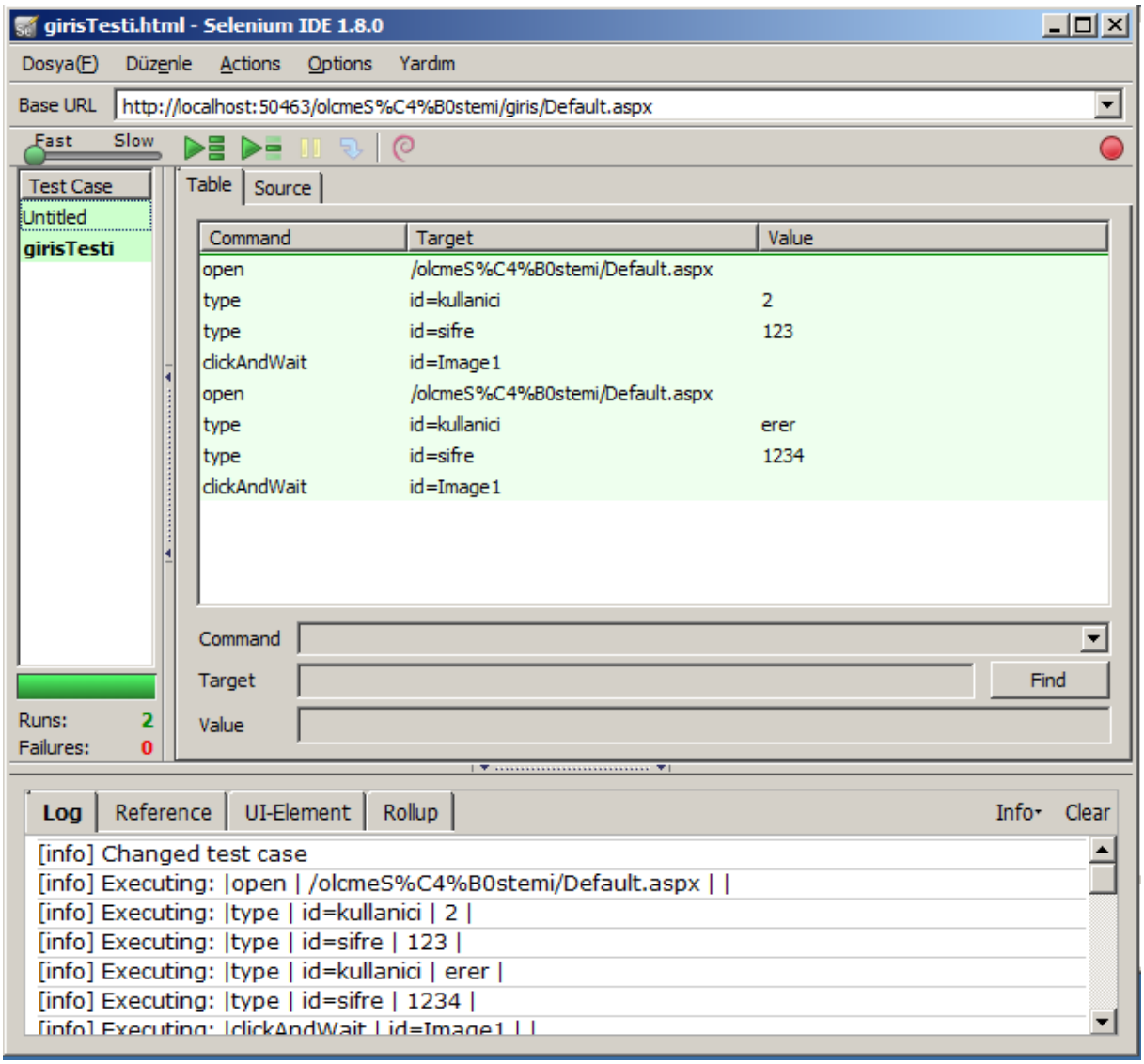
Liste 1**:** Kullanıcı girişi için oluşturulan script

```
<link rel="selenium.base"
href="http://localhost:50463/olcmeS%25C4%25B0stemi/giris/Default.aspx" />
<title>girisTesti</title>
</head>
<body>
```

```
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">girisTesti</td></tr>
</thead><tbody>
<tr>
        <td>open</td>
        <td>/olcmeS%C4%B0stemi/Default.aspx</td>
        <td></td>
</tr>
<tr>
        <td>type</td>
        <td>id=kullanici</td>
        <td>erer</td>
</tr>
<tr>
        <td>type</td>
        <td>id=sifre</td>
        <td>1234</td>
</tr>
<tr>
        <td>clickAndWait</td>
        <td>id=Image1</td>
        <td></td>
</tr>

</tbody></table>
</body>
</html>
```

Liste 1'deki test scripti 2 kullanıcı için yazılmıştır. Kod html formatında yazılmıştır. İlk satırdaki "selenium.base" referansı ile Selenium test aracının çalışması sağlanmıştır. Yazılan kodda her bilgi html tablo hücresi şeklinde belirtilmektedir. Tablonun ilk satırı script komut türünü ifade etmekte, diğer satır hücrelerinde ise script için gerekli veriler yer almaktadır. Yeni kullanıcı adı ve şifreler eklenirse daha çok kullanıcı için test işlemi ve bu sayede giriş için kullanılan kullanıcı adı ve şifre

bilgileri doğrulaması yapılmış olur. Şekil 5.1'de Selenium IDE programının bir ekran gerçekleştirim çıktısı görülmektedir.



Şekil 5.1. Kullanıcı adı ve şifre doğrulama ekranı

Çalıştırma işleminden sonra birinci kullanıcı adı ve şifre için giriş işlemi simülasyonu başarılı olur. Fakat ikinci sıradaki bilgiler yanlış olduğu için "click and wait" bölümünde kalınır.

Test scriptinin html formatında yazılabilmesi test işlemini kolaylaştırmaktadır. Ancak kullanıcı giriş işlemi için hazırlanan test scriptinde birden fazla kullanıcı için test yapabilmek için giriş sayfasına dönüş scripti yazmak zorundayız. Bu

zorunluluğun test verileri arttırıldıkça test işlemini yavaşlattığı gözlemlenmektedir. Ayrıca yazılan "test case" adımları herhangi bir "test suite" projesine eklenip kullanılabilmektedir. Selenium IDE'nin test kayıt fonksiyonu sayesinde ayrıca script yazmadan da kayıt işlemi yapılabilmektedir. Bunun için test kayıt URL belirtimi yapmak ve "record" butonuna basmak yeterlidir. Dolayısıyla otomatik olarak test işlemini gerçekleştirmek mümkün olur.

5.2.2. Soru ekleme sayfasının test edilmesi

Soru ekleme sayfası için yazdığımız script Liste 2'de görülmektedir.

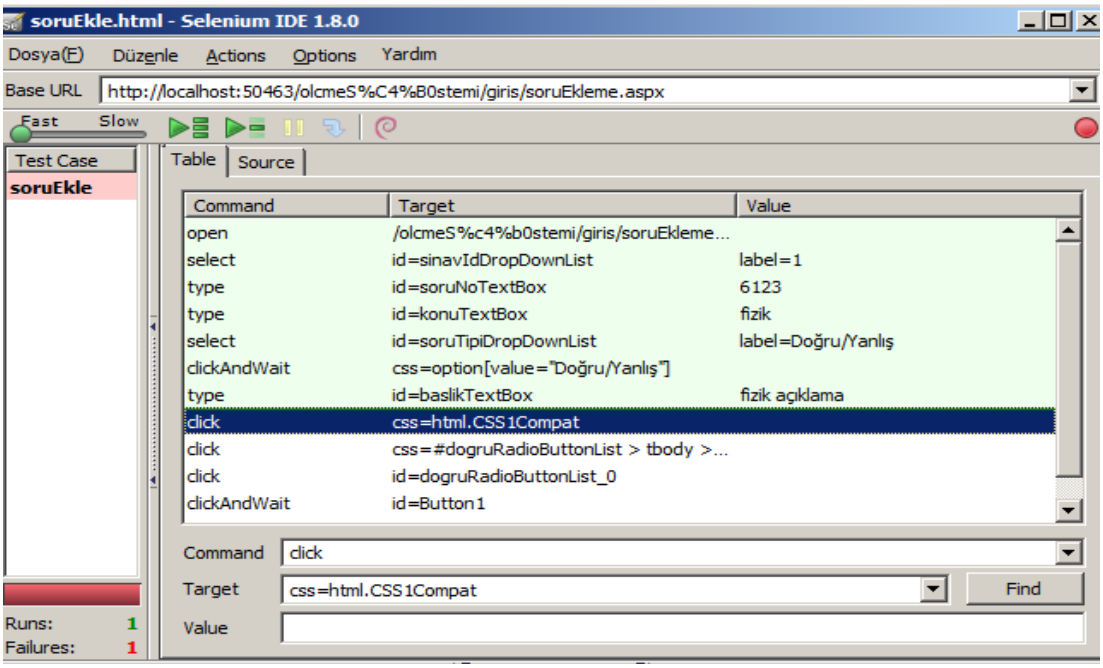
Liste 2: Soru ekleme işlemi test scripti

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base" href="http://localhost:50463/olcmeS%25C4%25B0stemi/giris/soruEkleme.aspx" />
<title>soruEkle</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">soruEkle</td></tr>
</thead><tbody>
<tr>
        <td>open</td>
        <td>/olcmeS%c4%b0stemi/giris/soruEkleme.aspx</td>
        <td></td>
</tr>
<tr>
        <td>click</td>
        <td>css=html.CSS1Compat</td>
        <td></td>
</tr>
<tr>
        <td>select</td>
```

```
            <td>id=sinavIdDropDownList</td>
            <td>label=2</td>
</tr>
<tr>
            <td>click</td>
            <td>id=konuTextBox</td>
            <td></td>
</tr>
<tr>
            <td>click</td>
            <td>css=html.CSS1Compat</td>
            <td></td>
</tr>
<tr>
            <td>click</td>
            <td>id=dogruRadioButtonList_1</td>
            <td></td>
</tr>
<tr>
            <td>click</td>
            <td>id=dogruRadioButtonList_0</td>
            <td></td>
</tr>
<tr>
            <td>clickAndWait</td>
            <td>id=Button1</td>
            <td></td>
</tr>

</tbody></table>
</body>
</html>
```

Şekil 5.2'de görüldüğü gibi, soru ekleme sayfasında doğru/yanlış seçim radioButtonList üzerinde id/css uyumsuzluğu hatası bulunmuştur. TextBox Ckeditor üzerine bağlandığından CkEditor bulunamamaktadır. Hatayı gidermek için CKEditor kod bölümü kaldırıldı. Yapılan her işlem Liste 2'de görüldüğü gibi Html kodları şeklinde işlenmektedir. İşlemi "tr" sayfa satır kodları göstermektedir. "td"

bölümlerinde ise test edilen elemana ait id bilgisi ve hangi test işleminin yapıldığı görülmektedir.



Şekil 5.2. Soru ekleme sayfası Selenium IDE test işlemi ekran görüntüsü

Yeni scriptin tablo görünümü Tablo 5.1'deki gibidir.

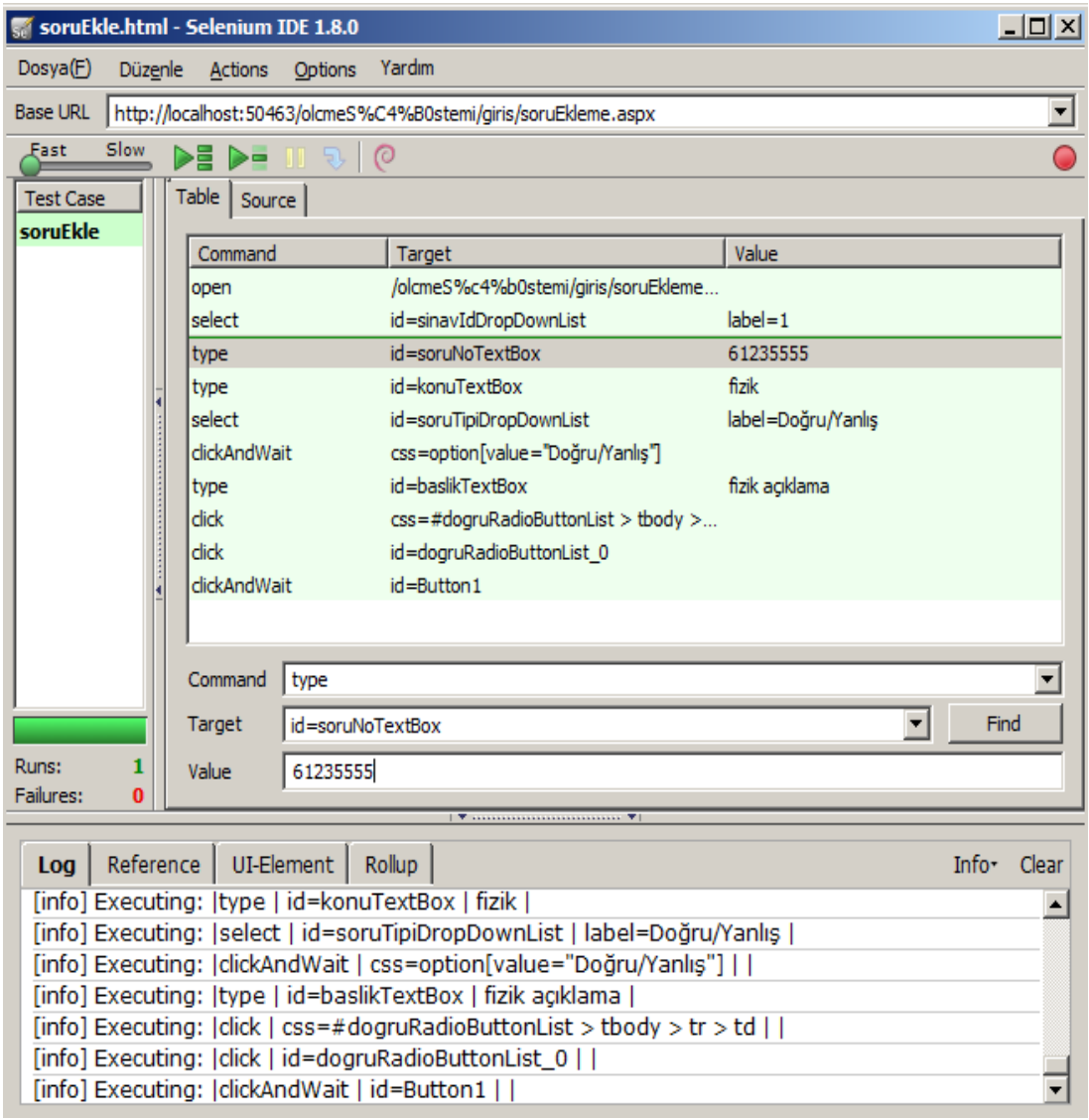
Tablo 5.1. Web sayfa elemanları tablosu

soruEkle		
open	/olcmeS%c4%b0stemi/giris/soruEkleme.aspx	
select	id=sinavIdDropDownList	label=1
type	id=soruNoTextBox	61235555
type	id=konuTextBox	fizik
select	id=soruTipiDropDownList	label=Doğru/Yanlış
clickAndWait	css=option[value="Doğru/Yanlış"]	
type	id=baslikTextBox	fizik açıklama
click	css=#dogruRadioButtonList > tbody > tr > td	
click	id=dogruRadioButtonList_0	
clickAndWait	id=Button1	

Yazılım mühendisliğinde kullanılan değişkenlerin ve yazılım nesnelerinin adlandırılması kodun yönetilmesi açısından önemlidir. Selenium IDE ile yapılan test kayıt işleminde kullanılan web sayfa elemanları Tablo 5.1'de listelenmektedir. Bu

sayede izlediğimiz kodlama standardına uymayan elemanların yazım hatalarını da düzeltmemiz kolaylaşmaktadır.

Test işlemi CKEditor bölümünün çıkarılmasıyla başarılı bir şekilde gerçekleştirilmiştir.



Şekil 5.4. Soru ekleme sayfası Selenium IDE test işlemi ekran görüntüsü 2

Yazdığımız scripti C# koduna çevirdiğimizde Liste 3'teki gibi görünecektir. Bu C# kodu Visual Studio 2010 Test projelerinde çalıştırılabilir.

Liste 3: Soru ekleme işlemi C# test scripti

```
using System;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;
using NUnit.Framework;
using OpenQA.Selenium;
using OpenQA.Selenium.Firefox;
using OpenQA.Selenium.Support.UI;

namespace SeleniumTests
{
  [TestFixture]
  public class Kod
  {
    private IWebDriver driver;
    private StringBuilder verificationErrors;
    private string baseURL;

    [SetUp]
    public void SetupTest()
    {
      driver = new FirefoxDriver();
      baseURL = "http://localhost:50463/olcmeS%C4%B0stemi/giris/soruEkleme.aspx";
      verificationErrors = new StringBuilder();
    }

    [TearDown]
    public void TeardownTest()
    {
      try
      {
        driver.Quit();
      }
      catch (Exception)
      {
        // Ignore errors if unable to close the browser
      }
      Assert.AreEqual("", verificationErrors.ToString());
    }
```

```
    [Test]
    public void TheKodTest()
    {
      driver.Navigate().GoToUrl(baseURL + "/olcmeS%c4%b0stemi/giris/soruEkleme.aspx");
      new SelectElement(driver.FindElement(By.Id("sinavIdDropDownList"))).SelectByText("1");
      driver.FindElement(By.Id("soruNoTextBox")).Clear();
      driver.FindElement(By.Id("soruNoTextBox")).SendKeys("6123555522");
      driver.FindElement(By.Id("konuTextBox")).Clear();
      driver.FindElement(By.Id("konuTextBox")).SendKeys("fizik");
      new
SelectElement(driver.FindElement(By.Id("soruTipiDropDownList"))).SelectByText("Doğru/Yanlış");
      driver.FindElement(By.CssSelector("option[value=\"Doğru/Yanlış\"]")).Click();
      driver.FindElement(By.Id("baslikTextBox")).Clear();
      driver.FindElement(By.Id("baslikTextBox")).SendKeys("fizik açıklama");
      driver.FindElement(By.CssSelector("#dogruRadioButtonList > tbody > tr > td")).Click();
      driver.FindElement(By.Id("dogruRadioButtonList_0")).Click();
      driver.FindElement(By.Id("Button1")).Click();
    }
    private bool IsElementPresent(By by)
    {
      try
      {
        driver.FindElement(by);
        return true;
      }
      catch (NoSuchElementException)
      {
        return false;
      }
    }
  }
}
```

Liste 3'te görülen C# test kodu Selenium IDE ile elde edilen otomatik test kaydı sonucu elde edilmiştir. Oluşturulan test kodu Java koda da dönüştürülebilmektedir. Çeşitli kod oluşturma seçeneklerinin olması, farklı yazılım test araçlarında kodların denenmesine olanak sağlamaktadır. Örneğin C# test kodları Watin açık kaynak kodlu yazılım test aracında ve Java kodları JUnit test aracında kullanılabilmektedir.

5.2.3. Sayfa linklerinin test edilmesi

Liste 4'deki test scripti sayesinde sayfa link ve butonlarının çalışıp çalışmadıkları test edilmektedir.

Liste 4: Sayfa link test scripti

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base"
href="http://localhost:50463/olcmeS%25C4%25B0stemi/giris/Default.aspx" />
<title>linkTest</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">linkTest</td></tr>
</thead><tbody>
<tr>
        <td>click</td>
        <td>incele</td>
        <td>click</td>
        <td>soruEkle</td>
        <td>click</td>
        <td>soruBankasi</td>
        <td>click</td>
        <td>performans</td>
        <td>click</td>
        <td>sifreDegistir</td>
        <td>click</td>
        <td>sinavOlustur</td>
        <td>click</td>
```

```
            <td>kullaniciGuncelle</td>

            <td></td>
</tr>

</tbody></table>
</body>
</html>
```

Tablo 5.2. Selenium IDE Test sonuçları tablosu

İşlev Testi Butonu	**Test Sonucu**
İncele	✔
Soru ekle	✔
Soru bankası	✔
Performans	✔
Şifre değiştir	✔
Sınav oluştur	✔
Kullanıcı güncelle	✔

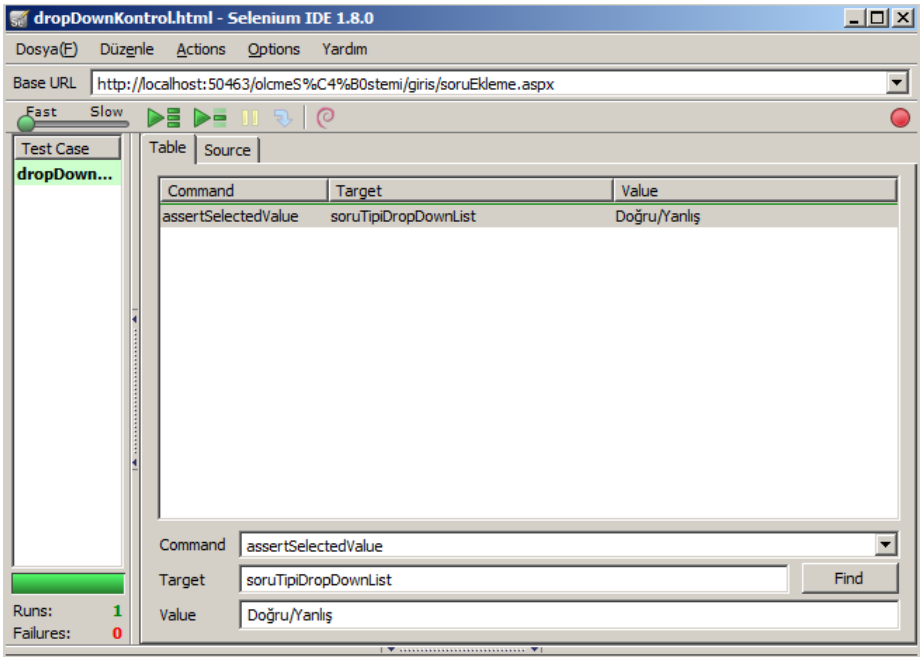
5.2.4. Soru ekleme sayfası seçili değer kontrolü

Web sayfasında kullanılan DropDownList elemanın beklenen değeri ile asıl değerinin aynı olup olmadığını kontrol eder. Eğer seçilen değer ile istenen değer uyuşmuyorsa “Actual value did not match” şeklinde bir mesaj alınır. Soru ekleme sayfamızda soru seçimi için yanlış eklenen bir değeri tespit etmemizi sağlar. Kontrol için yazılan testi scripti Liste 5’teki gibidir.

Liste 5: DropDownList test scripti

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
```

```
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link                                                    rel="selenium.base"
href="http://localhost:50463/olcmeS%25C4%25B0stemi/giris/soruEkleme.aspx" />
<title>dropDownKontrol</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">dropDownKontrol</td></tr>
</thead><tbody>
<tr>
        <td>assertSelectedValue</td>
        <td>soruTipiDropDownList</td>
        <td>Doğru/Yanlış</td>
</tr>

</tbody></table>
</body></html>
```



Şekil 5.5. DropDownList kontrolü Selenium IDE ekran görüntüsü

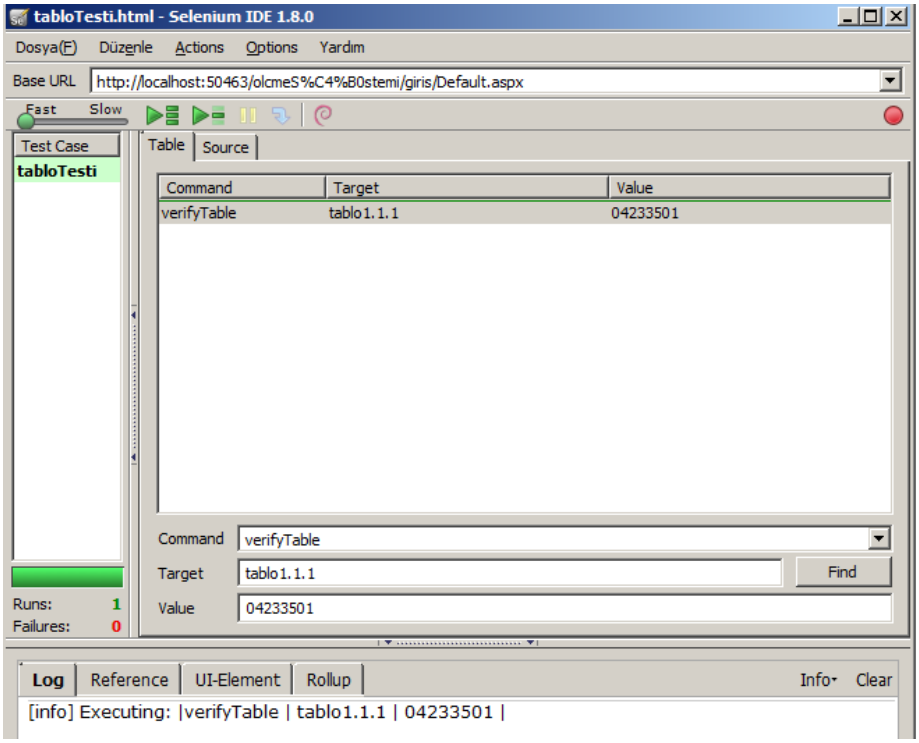
5.2.5. Tablo verisi kontrolü

Liste 6'daki testte giriş/Default.aspx sayfasındaki "tablo1" id sine sahip tablo değer testi yapılmıştır. Yazılan script Liste 6'da verilmektedir. "tablo1.1.1" yazımında tablo1 in 1. Satırının 1. Sütun elemanı kastedilmektedir. Soru Bankası sayfasında kullanılan tablolardaki verilerin test edilebilmesi için "verifyTable" komutu kullanılmıştır.

Liste 6: Tablo veri değer kontrol scripti

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head profile="http://selenium-ide.openqa.org/profiles/test-case">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<link rel="selenium.base"
href="http://localhost:50463/olcmeS%25C4%25B0stemi/giris/Default.aspx" />
<title>tabloTesti</title>
</head>
<body>
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">tabloTesti</td></tr>
</thead><tbody>
<tr>
        <td>verifyTable</td>
        <td>tablo1.1.1</td>
        <td>04233501</td>
</tr>

</tbody></table>
</body>
</html>
```

Şekil 5.6'da kullanıcı girişinde kullanılan verilerin tablo bilgilerinin doğruluğu test edilmiştir. Çalıştırılan test komut adedi bir adet ve başarılı test sayısı bir adet olarak elde edilmiştir.

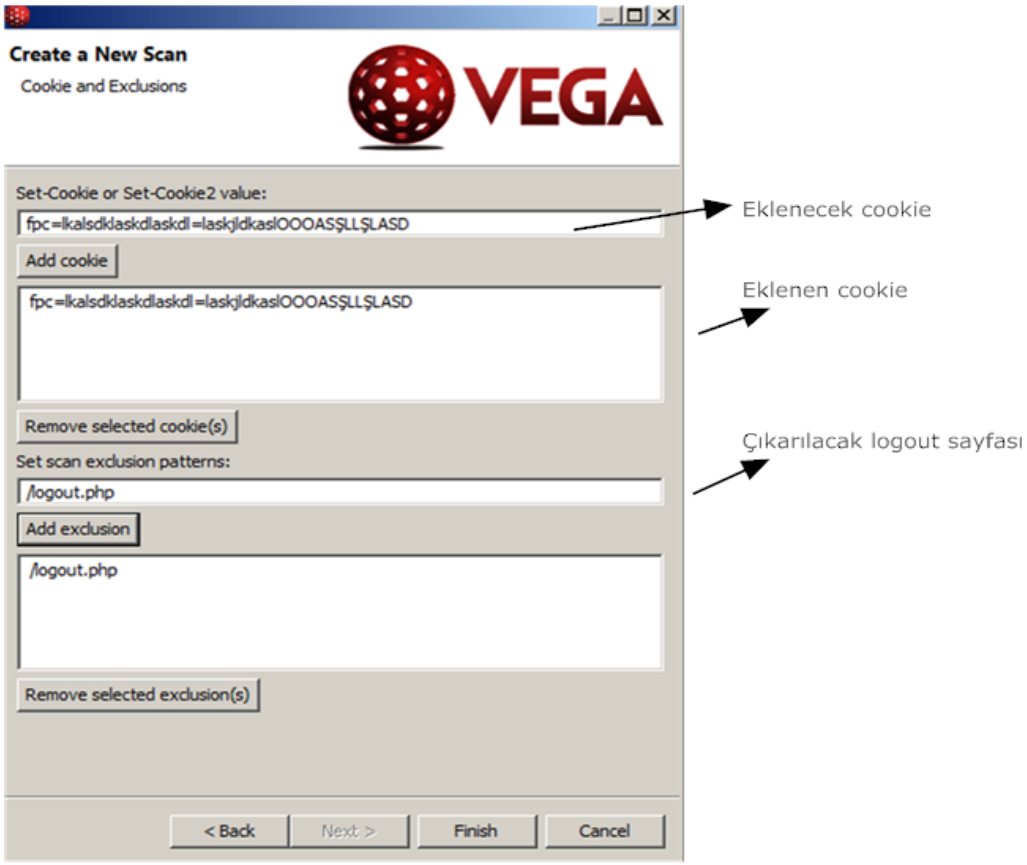


Şekil 5.6. Tablo testi Selenium IDE ekran görüntüsü

5.3. Vega Subgraph İle Yapılan Testler

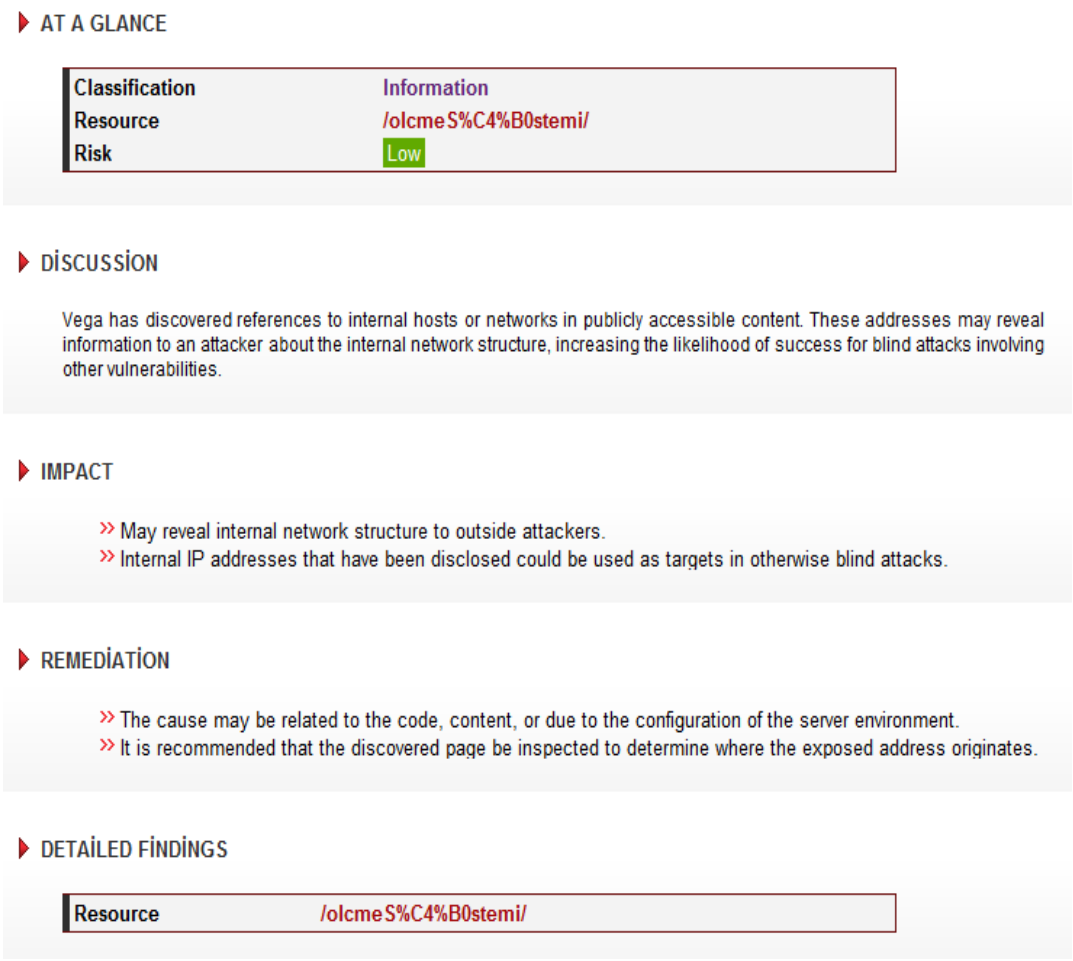
Vega Eclipse tabanlı web sayfalarının güvenlik açıklarının bulunmasına yardımcı olan bir yazılımdır [26]. Vega ile yaptığımız testler güvenlik açık tespit işlemleri olduğu için yazılım test tekniklerinden beyaz kutu test tekniği sınıfına girmektedir. Bulunan açıkların kapatılması için yapılan kod üzerindeki işlemler de statik test metodu olarak adlandırılabilir. İstenilen URL adresleri tek tek girilip test edilebildiği gibi Vega Proxy yardımıyla yapılan işlemler eş zamanlı olarak test edilebilir. Alınan uyarılar "Düşük", "Orta" ve "Yüksek" olarak sınıflandırılmaktadır. ÖDS'nin testi için aşağıdaki adımları yapıyoruz.

- Yeni bir tarama başlatıp sayfaya cookie ekliyoruz. Vega'nın kendi kendini loglamaması için "/logout.php" adresini ekliyoruz (Şekil 5.7.).



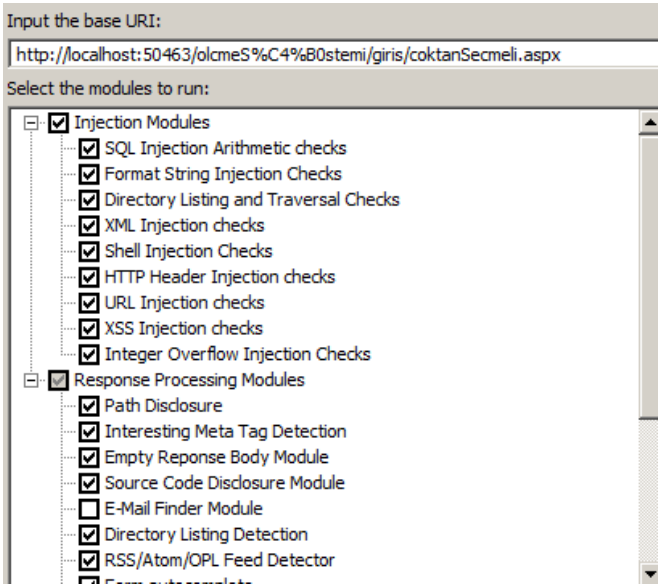
Şekil 5.7. Vega Cookie ekleme ekranı

ÖDS'nin giriş sayfasında "Low" düzeyinde klasör altında link gizleme yapılmamasından kaynaklanan bir güvenlik açığı bulunmuştur. Hata bilgileri dört başlık altında bildirilmektedir. Bu başlıklar "AT A GLANCE", "DISCUSSION", "IMPACT", "REMEDIATION" ve "DETAILED FINDINS" olarak tanımlanmıştır. Hata risk seviyesi ve kaynak bilgileri "AT A GLANCE" başlığı altında Şekil 5.8'deki gibi elde edilmiştir. Diğer başlıklar hata ile ilgili standart bildirimlerdir. Bu başlıklar altında genel olarak hatanın çıkış sebebi, sistem üzerinde oluşturabileceği etkiler ve iyileştirme için yapılabilecekler açıklanmıştır.



Şekil 5.8. Vega hata açıklama ekranı

Çoktan seçmeli soru sayfası için "injection" seçimi de yapıp veri tabanı güvenlik açıkları da bulunabilmektedir. "Injection" açıkları için dokuz adet seçenek vardır. Sistem açıklarının tespit süresini azaltmak için seçilen seçenek sayısı azaltılabilir. Ancak sistem açıklarının tam olarak tespit edilebilmesi için tüm seçeneklerin seçilmesi faydalı olacaktır. Şekil 5.9'da sistem üzerinde "coktanSecmeli.aspx" sayfasında yapılan güvenlik açık tespiti için Vega Subgraph test yazılımı üzerinden işaretlenen seçenekler gösterilmiştir. Proxy ayarları yapıldıktan sonra istenirse bu seçim yapılan tüm işlemler üzerinde test edilebilmektedir.



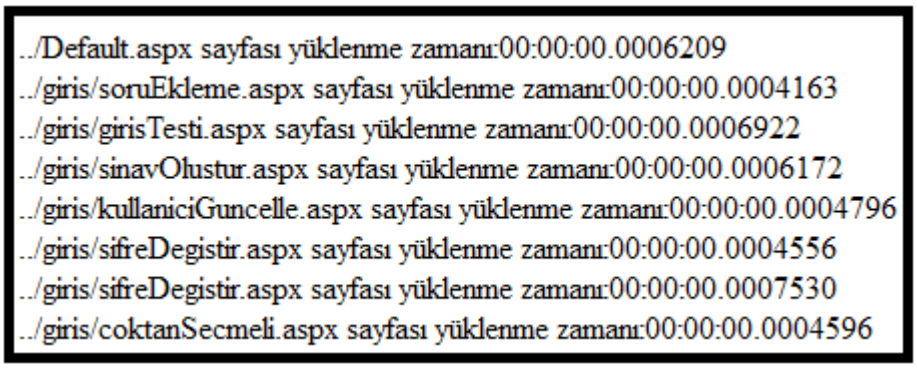
Şekil 5.9. Vega güvenlik açıkları seçenek ekranı

Vega Subgraph web güvenlik test aracı sayesinde ÖDS'deki güvenlik açıklarını kolaylıkla tespit edebildik. Bunun yanında tasarladığımız sistemin veri tabanı güvenlik açıklarını da varsa çözmüş olduk. Uzaktan eğitimde kullanılan ölçme değerlendirme güvenliği son derece önemlidir. Güvenlik açıklarının bulunup yok edilmesi daha sağlıklı bir ölçme değerlendirme sistemi tasarlamamızı sağlamıştır.

İç içe iki adet klasör olduğu için "Low" düzeyinde iki adet uyarı alınmıştır. Linklerin gizlenmemesi durumunda kör saldırılara maruz kalınabilir. Sistem kodlaması Asp.net ile geliştirildiği için sql injection bulunamamıştır.

5.4. Sayfa Yükleme Zamanı Testleri

Performans testi için daha önceki bölümlerde bahsedilen araçlar kullanılabileceği gibi C# test kodları da yazılarak performans ölçümü yapılabilmektedir. Hesaplamalar yapılırken nesne yönelimli programlamanın özelliklerinden faydalanılmıştır [33]. Sayfa yükleme zamanı testi sayfalara ait ulaşan ilk bayt ile son bayt arasında geçen zaman hesaplanarak bulunabilir. ÖDS üzerinde yapılan sayfa yükleme zamanı sonuçları Şekil 5.10'da gösterilmektedir.



Şekil 5.10. Vega güvenlik açıkları seçenek ekranı

Sayfa yüklenme zamanları nanosaniye ve milisaniye cinsinden Şekil 5.10'da performans.aspx sayfasında hesaplanmıştır. Kullanılan fonksiyon Liste 7'deki gibidir:

Liste 7: Sayfa yükleme zamanı kod listesi

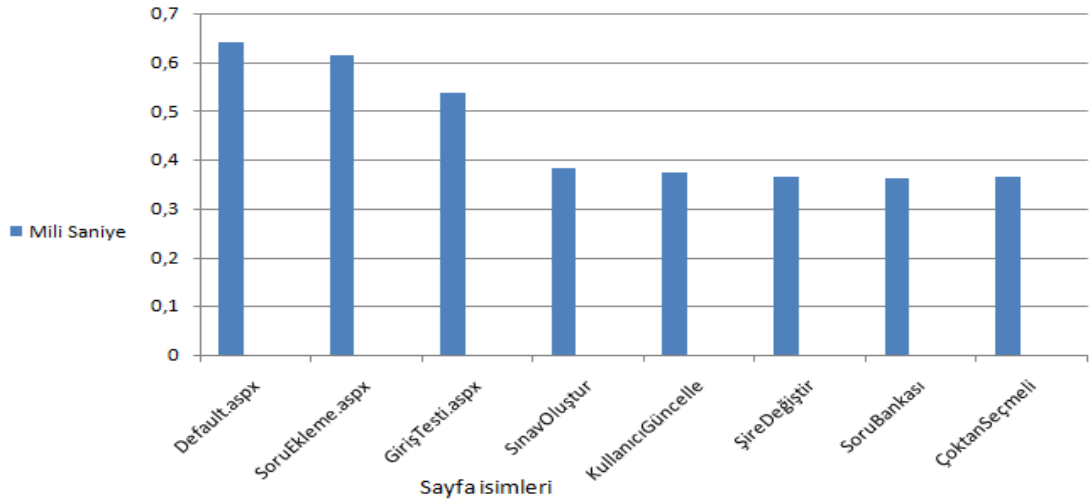
```
public Stopwatch hesapla(string yol)
   {

      WebClient client = new WebClient();
      client.Headers.Add("user-agent", "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.2; .NET CLR
1.0.3705;)");
      //dördüncü sayfa için kod
      Stopwatch stopwatch = new Stopwatch();
      stopwatch.Start();

      Stream data = client.OpenRead(Server.MapPath(yol));
      StreamReader reader = new StreamReader(data);
      string s = reader.ReadToEnd();

      stopwatch.Stop();
      Response.Write("</br>" +yol+" sayfası yüklenme zamanı:" + stopwatch.Elapsed.ToString());

      data.Close();
      reader.Close();
```

```
    return stopwatch;

}
```

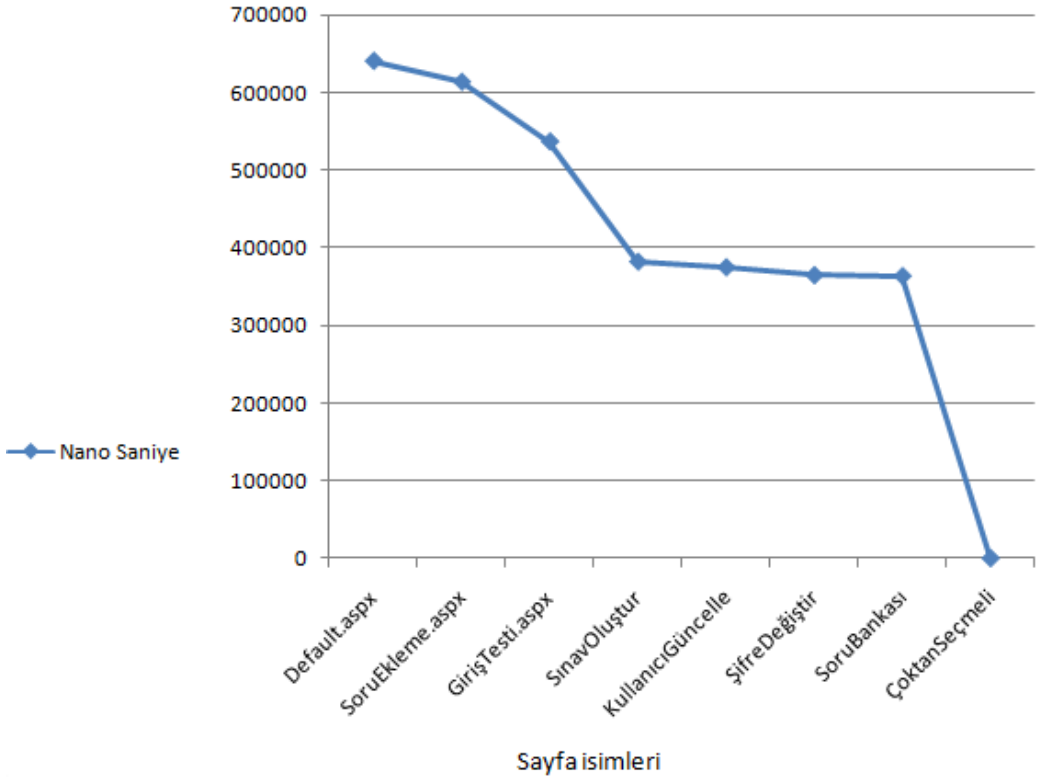
Liste 7'de kullandığımız fonksiyon "Stopwatch" türünde bir değer döndürmektedir. Sayfa yükleme zamanı hesaplanırken ulaşan ilk bayttan son bayta kadar geçen süre hesaplanmıştır. "Stopwatch" nesnesinin oluşturulan örneğine "yol" değişkeni ile sayfanın yolu atanmıştır. Okunan veriler bu yol üzerinden okunmaktadır. Sayfanın "Page_load" fonksiyonu içinde aşağıdaki gibi çağrılmıştır.

```
//dördüncü sayfa için kod
 string yol4 = "../giris/sinavOlustur.aspx"; Stopwatch deger4;
 deger4 = hesapla(yol4);
 //dördüncü sayfa için kod
```

Elde edilen sonuçlar rapor.xls dosyasına yazdırılmıştır. Şekil 5.11'de milisaniye cinsinden ve Şekil 5.12'de nanosaniye cinsinden sayfa yüklenme zaman grafikleri görülmektedir.



Şekil 5.11. Sayfa yüklenme zamanı grafiği (ms)



Şekil 5.12. Sayfa yüklenme zamanı grafiği (ns)

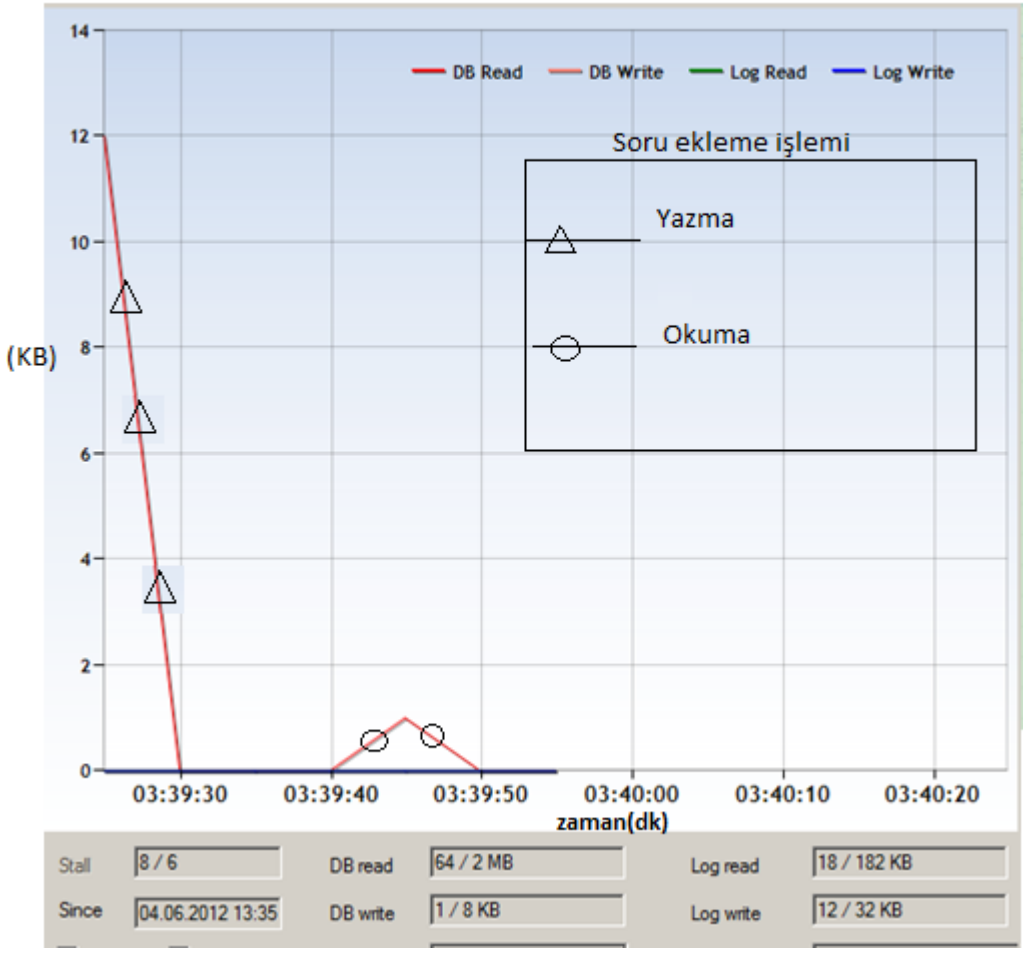
Sayfa yükleme zamanlarının test edilmesi yazılım test tekniklerinden performans test bölümüne girmektedir. Grafikten elde ettiğimiz sonuçlar test ölçütleri olarak nitelendirilebilir. Grafikten okunan değerlere göre geç yüklenen sayfalar üzerinde beyaz kutu test uygulanabilir. Böylece gereksiz değişken kullanımları, karar mekanizmalarının azaltılması ve yapılan işlemleri fonksiyonlar yardımıyla çözmek gibi işlemlerle yükleme zamanları düşürülmüş olur.

5.5. Veri Tabanı Performans Testleri

5.5.1. SqlMon ile yapılan testler

SqlMon aracı ile yapılan "olcme" isimli veri tabanımızın performans grafiği "soruEkleme.aspx" sayfası için Şekil 5.13'teki gibidir. Soru ekleme işlemleri sırasından okunan veriler Şekil 5.13'de üçgen işareti ve yazılan veriler daire işareti kullanılarak gösterilmiştir. Grafik KB cinsinden değerleri vermektedir. Veri tabanı

Log dosyasına yazılan veriler için yeşil çizgi ve Log dosyasına yazılan veriler için mavi çizgi kullanılmaktadır. Soru ekleme işlemi için Log dosyası kullanılmadığından grafikte bu işlemlerle ilgili bir bilgi gözükmemektedir.

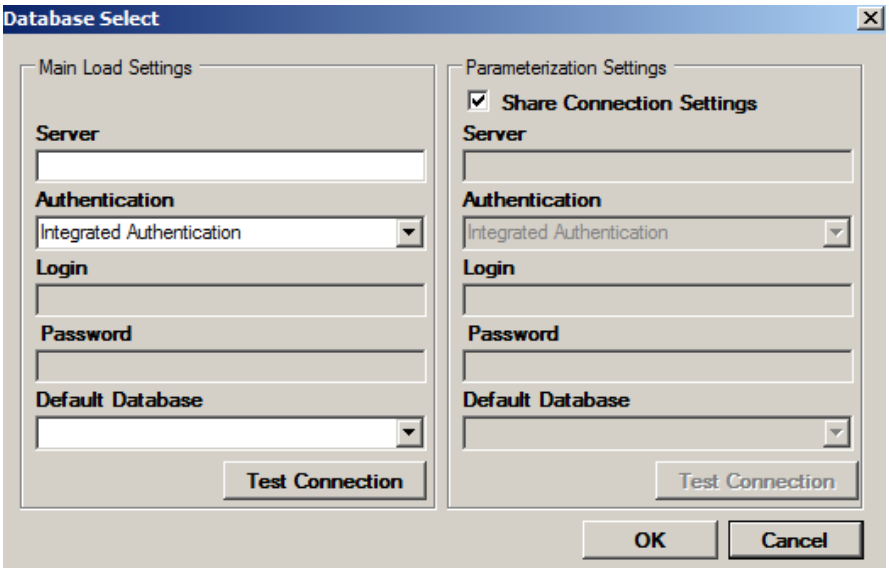


Şekil 5.13. Veri tabanı performans test grafiği

5.5.2. SqlQueryStress ile yapılan testler

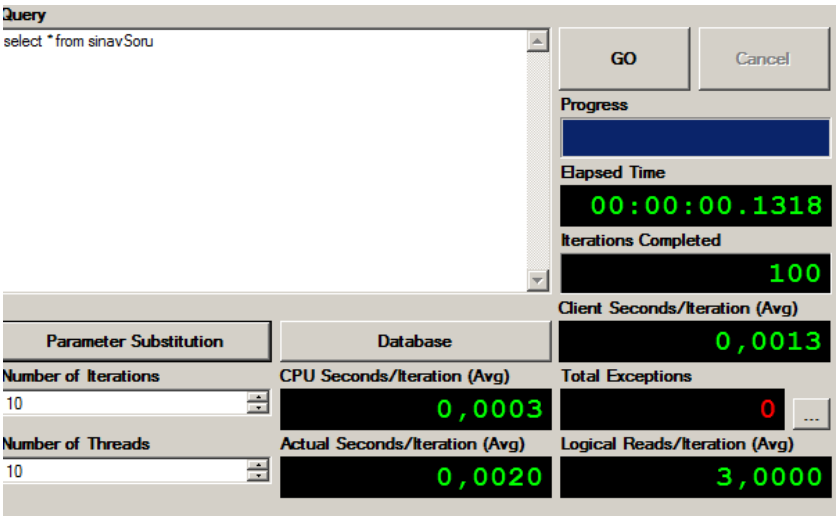
SqlQueryStress açık kaynak kodlu bir performans test aracıdır. Bu araç kullanılarak T-SQL sorgu tabanlı veri tabanı testi yapılabilmektedir.

"Database" bölümünden test edilecek veri tabanı ayarları yapılır. Ayarlar ekranı Şekil 5.14'te görülmektedir.



Şekil 5.14. Sql QueryStress veri tabanı ayarları ekranı

"Query" bölümüne SQL sorgusu yazılır. "Number of Iteration " bölümüne sorgunun kaç defa çalıştırılacağı belirtilir. Sorgunun kaç kullanıcı için çalıştırılacağı ise "Number of Threads" bölümünde belirtilir. Kullanıcı sayısı 200 ile sınırlandırılmıştır. "Client Seconds/Iteration" bölümünde tüm iterasyonların ortalama zamanı görüntülenir. Çalıştırılan sorgu Şekil 5.15'da görülmektedir.



Şekil 5.15. Sql QueryStress sorgu sonuç ekranı

Ölçme değerlendirme sistemimiz üzerinde yaptığımız 10 kullanıcı ve 10 iterasyon için bazı test sorguları ve sonuçları Tablo 5.3'te görülmektedir:

Tablo 5.3. Sorgu sonuçları değerler tablosu

Sorgu	**Geçen Zaman**	**İterasyon adedi**	**Kullanıcı Adedi**	**Ortalama iterasyon zamanı**
select * from sinavSoru	00:00:00.1318	10	10	0,0013
Select * From tumPersonel Where kullaniciAdi='mmaruf' and sifre='123'	00:00:00.0693	10	10	0,0007
SELECT es1,es2,es3,es4,es5,es6,es7,es8,es9,esCevap1,esCevap2,esCevap3,esCevap4,esCevap5,esCevap6,esCevap7,esCevap8,esCevap9 from sinavSoru where tip='Eşleştirme'	00:00:00.1112	10	10	0,0032
Select id,soruBaslik,soruMetin, sec1,sec2,sec3,sec4,cevap from sinavSoru where tip='Çoktan Seçmeli'	00:00:00.0966	10	10	0,0016
Select * from ogrenci	00:00:00.3720	10	10	0,0105
Select * from yonetici	00:00:00.1406	10	10	0,0066
Select * from ogretmen	00:00:00.1259	10	10	0,0041
Select from sinav	00:00:00.0908	10	10	0,0000

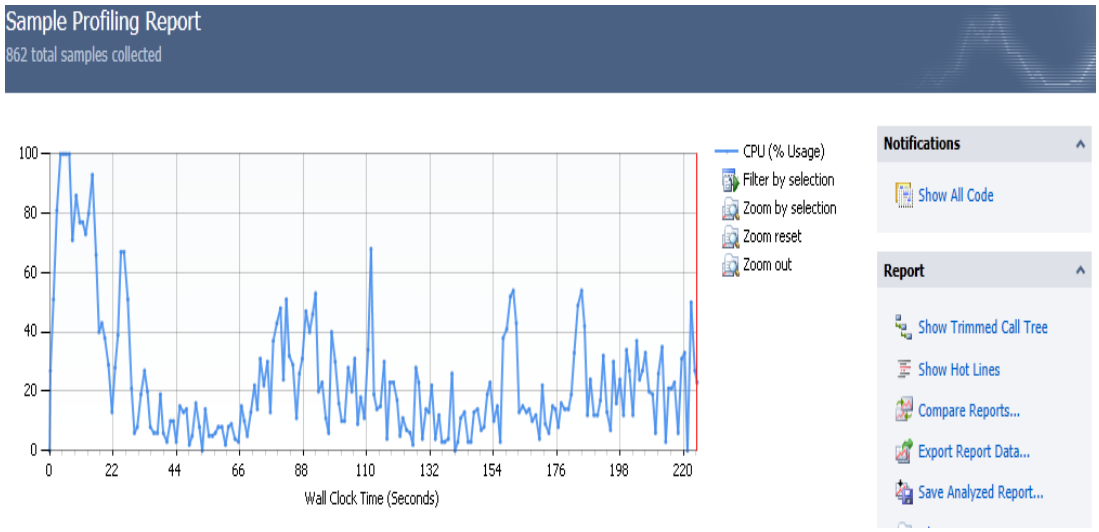
Tablo 5.3'teki performans test sonuçlarımıza göre ölçme değerlendirme sisteminde test edilen tablolardaki alan sayısı, kullanıcı sayısı ve iterasyon adedi arttıkça sistem cevap süresinin uzadığı gözlemlenmektedir. Bu nedenle veri tabanındaki tablolardan

sorgulama yaparken belirli alanlar çekilmeli her tablodaki alan sayısı olabildiğince minimuma indirilmelidir.

5.6. PerformanceAnalyzer İle Yapılan Testler

PerformanceAnalyzer Visual Studio Ultimate ve Premium ile gelen bir özelliktir. Bu özellik sayesinde projedeki sınıfların ve fonksiyonların CPU performansları ölçülebilir. Ölçme değerlendirme sistemimiz için yaptığımız performans testi sonuçları Şekil 5.16'daki gibidir:

Şekil 5.16'te görülen grafikte proje çalışırken kullanılan sınıf ve fonksiyonlar için CPU kullanım yüzdeleri verilmektedir. 862 (sınıf, fonksiyon) örnek için çalışma sonucu görülmektedir. Sistem ilk çalıştığında CPU %100 kullanıma kadar çıkmakta daha sonra kullanılan fonksiyon ve sınıflara göre değişkenlik göstermektedir.



Şekil 5.16. İşlemler performans analiz ekranı

"Hot Path" bölümünde "Inclusive Samples" yüzdesi yüksek olan sınıflar sistemi en çok zorlayan sınıflardır. "Exclusive Samples" bölümünde ise performans darboğazı yaratan fonksiyonlar en yüksek yüzdeye sahiptir.

Tablo 5.4. Sayfa CPU kullanım oranları tablosu

Hot Path

The most expensive call path based on sample counts

Function Name	Inclusive Samples %	Exclusive Samples %
ASP.giris_sorubankasi_aspx.ProcessRequest(class System.Web.HttpContext)	6,03	0,00
ASP.giris_coktansecmeli_aspx.ProcessRequest(class System.Web.HttpContext)	3,83	0,00
ASP.giris_kullaniciguncelle_aspx.ProcessRequest(class System.Web.HttpContext)	3,60	0,00
ASP.giris_default_aspx.ProcessRequest(class System.Web.HttpContext)	2,90	0,00
ASP.giris_soruİncele_aspx.ProcessRequest(class System.Web.HttpContext)	1,86	0,00

Related Views: Call Tree Functions

Tablo 5.5'te en çok işleve sahip fonksiyonlar "Exclusive Samples" bölümünde en yüksek yüzdeye sahiptir. Sistemimiz web tabanlı ve veri tabanı işlemlerine dayandığı için "HttpContext" ve "SqlConnection" işlemlerinin payları fazladır.

Tablo 5.5. Sınıf CPU kullanım oranları tablosu

Functions Doing Most Individual Work

Functions with the most exclusive samples taken

Name	Exclusive Samples %
System.Web.UI.Page.ProcessRequest(class System.Web.HttpContext)	43,74
[clr.dll]	33,76
System.Data.SqlClient.SqlConnection..ctor(string)	4,18
System.Data.SqlClient.SqlConnection.Open()	3,36
System.Data.Common.DbDataAdapter.Fill(class System.Data.DataSet)	2,67

Web sayfalarının çalışması için temel sınıf olan "HttpContext" sınıfının her sayfada çalıştığı için yüzdesinin yüksek olduğu görülmektedir. Sayfa üzerindeki veriler veri tabanı vasıtasıyla getirildiği için "SqlConnection" sınıfının da yüksek CPU kullanım yüzdesine sahip olduğu görülmektedir. Tablo 5.6'da kullanılan tüm fonksiyonlara ait CPU kullanım yüzdeleri görülmektedir. Ayrıca kullanım yüzdelerine ait ağaç görünümü de oluşturulabilmektedir. Oluşturulan raporlar CVS ve XML formatında projeden başka bir ortama da aktarılabilmektedir. Fonksiyon, işlem, başlık, iş parçası gibi seçenekler işaretlenerek raporun içeriği istenildiği şekilde düzenlenebilmektedir.

Tablo 5.6'da daha az yüzdeye sahip fonksiyonların "Exclusive Samples" ve "Inclusive Samples" oranları görülmektedir.

Tablo 5.6. Sınıf CPU kullanım oranı detay tablosu

Function Name	Inclusive Samples	Exclusive Samples	Inclusive Samples %	Exclusive Samples %
ASP.giris_default_aspx.__Build	4	0	0,42	0,00
ASP.giris_default_aspx.__Build	5	0	0,52	0,00
ASP.giris_default_aspx.Framev	5	0	0,52	0,00
ASP.giris_default_aspx.Proces	29	0	3,03	0,00
ASP.giris_dogruyanlis_aspx.__	2	0	0,21	0,00
ASP.giris_dogruyanlis_aspx.__	1	0	0,10	0,00
ASP.giris_dogruyanlis_aspx.__	1	0	0,10	0,00
ASP.giris_dogruyanlis_aspx.__	3	0	0,31	0,00
ASP.giris_dogruyanlis_aspx.Fr	3	0	0,31	0,00
ASP.giris_dogruyanlis_aspx.Pr	34	0	3,55	0,00
ASP.giris_giristesti_aspx.Proce	15	0	1,57	0,00
ASP.giris_kullaniciguncelle_asp	2	0	0,21	0,00
ASP.giris_kullaniciguncelle_asp	2	0	0,21	0,00
ASP.giris_kullaniciguncelle_asp	2	0	0,21	0,00
ASP.giris_kullaniciguncelle_asp	27	0	2,82	0,00
ASP.giris_raporlama_aspx.__B	19	0	1,99	0,00
ASP.giris_raporlama_aspx.__B	19	0	1,99	0,00
ASP.giris_raporlama_aspx.__B	20	0	2,09	0,00
ASP.giris_raporlama_aspx.Fran	20	0	2,09	0,00
ASP.giris_raporlama_aspx.getL	30	0	3,13	0,00
ASP.giris_raporlama_aspx.Pag	30	0	3,13	0,00
ASP.giris_raporlama_aspx.Proc	372	0	38,87	0,00
ASP.giris_sorubankasi_aspx.__	3	0	0,31	0,00
ASP.giris_sorubankasi_aspx.__	1	0	0,10	0,00
ASP.giris_sorubankasi_aspx.__	7	0	0,73	0,00
ASP.giris_sorubankasi_aspx.__	7	0	0,73	0,00

Tablo 5.6'dan da anlaşılacağı üzere sistemin başlatılma ve yüklenme zamanında CPU kullanımı yüksektir. Bu performans problemini aşmak için gereksiz sınıf ve fonksiyon yazımından kaçınılmalı az fonksiyonla çok iş hedeflenmelidir. Kullanılan fonksiyonlar "Dispose" metodu kullanılarak bellekten işlevi bittikten sonra atılmalıdır.

BÖLÜM 6. SONUÇLAR VE ÖNERİLER

Uzaktan eğitim yüksek öğretimin çeşitli öğrenim seviyelerinde uygulanmaktadır. Uzaktan eğitim için hem ücretli yazılımlar kullanılmakta hem de açık kaynak kodlu yazılımlar tercih edilmektedir. Lisans, yüksek lisans ve doktora seviyelerinde de uzaktan eğitim tez çalışmaları yapılmaktadır. Yapılan çalışmalarda yazılım testi araçlarının ve yazılım test tekniklerinin kullanılmaması sistem uyumsuzluklarına yol açmakta ve bakım maliyetleri bu nedenle artmaktadır. Yaptığımız çalışmada yazılım test tekniklerinin uygulandığı bir ölçme değerlendirme sistemi tasarlanmış, sonuçlar Tablo 6.1'deki gibi elde edilmiş ve açıklanmıştır.

Tablo 6.1. Test sonuçları tablosu

Test Aracı	**Yapılan Test**	**Sonuç**	**Değer**
Selenium IDE	Giriş sayfası testi	Başarılı işlem adedi	2
		Hata sayısı	0
Selenium IDE	Soru ekleme sayfası testi	Başarılı işlem adedi	0
		Hata sayısı	1
Selenium IDE	Kullanılan web elemanlarının işlev testi	Başarılı işlem adedi	1
		Hata sayısı	0
Selenium IDE	Tüm sayfa linklerinin test edilmesi	Başarılı işlem adedi	1
		Hata sayısı	0
Selenium IDE	Soru tipi seçimi testi	Başarılı işlem adedi	1
		Hata sayısı	0
Selenium IDE	Default.aspx tablo değer testi	Başarılı işlem adedi	1
		Hata sayısı	0
Vega Subgraph	Sistem güvenlik testi	Düşük seviyeli uyarı	2
		Sql injection	0
Visual Studio	Yükleme testi	Ortalama yükleme zamanı	00.00.00.0005618
SQLMON	Veri tabanı performans testi	Optimuma yakın kullanım	

SQLQueryStress	Veri tabanı sorgu testleri	Ortalama sorgu işlem zamanı	0.0025 sn
PerformanceAnalyzer	Cpu kullanım testi	En yüksek kullanım oranı	%6.03

Tablo 6.1'de sistem üzerinde yapılan testler ve sonuçları görülmektedir. Yapılan testler sonucunda sayfalardaki hatalar(failure) tespit edilmiş kodlar yeniden gözden geçirilmiştir. Kullanılmayan linkler sayfalardan silinmiştir. Test sonuçlarına göre yapılan değişikliklerle çevik bir yazılım elde edilmiştir [34]. Vega Subgraph aracı ile yapılan güvenlik testi sonucunda sayfalarda sql injection bulunamamıştır. Fakat sayfa erişimlerinin "\" işareti ile ayrılan klasörler üzerinden yapılması düşük seviyeli bir güvenlik açığıdır. Bu güvenlik açığının giderilmesi için url gizleme yöntemleri kullanılmalıdır. Veri tabanı testleri sonucunda daha uzun sorguların daha çok zaman aldığı görülmektedir. Bunun için veri tabanı işlemlerinde kod bölümünden daha çok tetikler(trigger) ve prosedür(stored procedure) kullanılmalıdır. Yapılan test işlemleri ayrı yazılımlar kullanılarak gerçekleştirilmiştir. Eğer güvenlik, performans ve hata analizlerin birlikte yapılabildiği bir yazılım geliştirilebilirse sistem kodlama ve sınama daha az zaman alacaktır. Yazılım testi çalışmaları günden güne artmaktadır [35]. Geliştirilen yazılımların gereksinimleri karşılayabilmesi ve istenen fonksiyonları yerine getirmesi için yazılım test araçları kullanılmaktadır. Yapılan bu çalışmada yazılım test teknikleri ve kullanım amaçlarına da değinilmiştir. Yapılan yazılım test işlemlerinin yapılacak yazılım test ve uzaktan eğitim çalışmalarına model olması beklenmektedir.

KAYNAKLAR

[1] İÇTEN, T., Uzaktan Eğitim Öğrencileri İçin Web Tabanlı Çevrimiçi Sınav Sistemi Uygulaması Geliştirilmesi, Y. Lisans, 1, 2006.

[2] NET-ClassR Learning Management System, https://online.metu.edu.tr/, (Erişim tarihi: Mayıs 2012).

[3] TASBASI, N, Aydın, A., Uzaktan Eğitimde Sakarya Üniversitesi Çözümleri, Açık ve Uzaktan Eğitim Sempozyumu, Eskişehir, 2002

[4] YENİLMEZ, E., CEBECİ, Z, KOÇAK, S., Çevrimiçi Sınav Sistemi Uygulamaları, Akademik Bilisim Konferansı'03, Çukurova Üniversitesi, Adana, 1-2, 2003.

[5] DEMİR, D., Endüstride yazılım testi ve kalite güvencesi etkinlikleri, I. Ulusal Yazılım Mühendisliği Sempozyumu**,** İzmir, 23-25 Ekim, 2003.

[6] TUNA, O., Yazılım geliştirme süreci ve mimari gösterime dayalı yazılım testi, Yüksek Lisans Tezi, Dokuz Eylül Üniversitesi Fen Bilimleri Enstitüsü, İzmir, 1, 2005.

[7] GÜRBÜZ, A., Yazılım Test Mühendisliği, Papatya Yayıncılık Eğitim, İstanbul, 31-34, 2010.

[8] IEEE Standards Association, Glossary of Software Engineering Terminology, http://standards.ieee.org/findstds/standard/610.12-1990.html, (Erişim tarihi: Mayıs 2012).

[9] SMITH, M.D., ROBSON, D.J, Object oriented programming the problems of validation, Software Maintenance, 1990. (ICSM '90) Proceedings. IEEE International Conference, CH2921-5/90/0000/0272, 1990.

[10] ANSI Std-1991, Standart glossary of software engineering terminology (ANSI), The institute of electrical and electronics engineers inc., 1991.

[11] MYERS, G., The Art of Software Testing, Wiley Interscience, ISBN: 471043281, 1979.

[12] GILL, A., Debugging haskell by observing intermediate data structures, In Proceedings of the 4th Haskell Workshop, Technical Report of the University

of Nottingham, 2000.

[13] LEWIS, E. W., Software Testing and Continuous Quality Improvement, 2nd ed., A CRC Pres Company, USA, 10, 60, 117-127, 129-155, 183-185, 230-256, 2005.

[14] BOURQUE, P. ve DUPUIS, R., Guide To The Software Engineering Body Of Knowledge, The Institute of Electrical and Electronics Engineers, USA, 5, 2004.

[15] DAVIS, S. ve arkadaşları, Software Testing Engineering With IBM Rational Functional Tester, Pearson Plc, USA, 54, 2010.

[16] Software Testing - Testing Tutorials, Testing Tools, Testing Softwares, Testing Jobs, Testing Techniques, http://www.onestoptesting.com, (Erişim tarihi: Mayıs 2012)..

[17] TİFTİK, N., ÖZTARAK, H., ERCEK, G. ve ÖZGÜN, S., Sistem/yazılım geliştirme sürecinde doğrulama faaliyetleri, III.Ulusal Yazılım Mühendisliği Sempozyumu, Ankara, 1-2, 2007.

[18] MULLER, T., GRAHAM, D., FRIEDENBERG, D. ve VEENDENDAL, E., International Software Testing Qualifications Board (ISTQB), Foundation Level Syllabus, USA, 10, 2007.

[19] SOMMERVILLE, I., Software Engineering, ISBN:0-201-39815-X, Addison Wesley, 2001.

[20] GARRIDO, J.M., Object Oriented Programming: From Problem Solving to Java, Charles River Media**,** USA, 19-23, 240-241, 2003.

[21] GÜNGÖREN, B., UML ile Nesne Tabanlı Çözümleme ve Tasarım, Seçkin Yayıncılık**,** Ankara, 87, 2005

[22] MUSTAFA, K. ve KHAN, R.A., Software Testing: Concept and Practices, India, Lucknow, 5-21, 227-228, 2007.

[23] O'REGAN, G., A Pratical Approach to Software Quality, Maple-Vail Book Manufacturing Group, Newyork, 71-77, 2002.

[24] FAGAN, M. E, Advanced in software inspection, IEEE Transactions on Software Engineering, Vol. SE-12, No.7, July, 1986.

[25] ZHANG, P., Software Engineering (ICSE), 34th International Conference 2012; 595:595–605, 2012.

[26] Subgraph Vega | Free and Open Source Web Application Vulnerability Scanner Site, http://www.subgraph.com/products.html, (Erişim tarihi: Mayıs 2012).

[27] Sakarya Üniversitesi Öğrenci İşleri Bilgi Sistemi, http://www.ido. sakarya.edu.tr / sertifika/default.asp?d=g1, 2005, (Erişim tarihi: Mayıs 2012).

[28] İnternette Güvenlik ,E-Ticaret Kütüphanesi, http://www.Uluslararasi egitim.com/uzak/geschich.asp, (Erişim tarihi: Mayıs 2012).

[29] YILMAZÇOBAN, S., DAMKACI, F., İnternet'in Eğitim Amaçlı Kullanılması**,** V. Türkiye'de Internet Konferansı, Ankara, 19, 1999.

[30] ATICI, B., Bilgisayar Destekli Asenkron İşbirlikli Öğrenme Yönteminin Sınıf Yönetimi Dersinde Öğrenci Başarısına Etkisi(F.Ü. Teknik Eğitim Fakültesi Örnegi), Yüksek Lisans Tezi, F.Ü. Sosyal Bilimler Enstitüsü**,** Elazığ, 10, 2000.

[31] ÇABUK, A., ERDOĞAN, S., Bilgisayar Destekli Tasarım ve Coğrafi Bilgi Sistemlerinin Kullanım olanaklarının Genişletilebilmesi için internet Tabanlı Egitim Modellerinden Yararlanılması, Akademik Bilişim 2001**,** Samsun**,** 12, 2001**.**

[32] GOODMAN, P., The Practical Implementation of Software Metrics, McGraw- Hill, New York, USA, 136, 1993.

[33] ERTEMEL, H.Ö., SELÇUK, Y.E, KALIPSIZ O., Nesneye yönelik sistemler için bir uyum ölçütü önerisi: Comıas, IV. Ulusal Yazılım Mühendisliği Sempozyumu**,** Ankara, 1, 2009.

[34] KAGATIKAR, M., G., Test2008 AGILITY IN TESTING, INDIA, 2008, 3.

[35] NAIK, K. ve TRIPATHY, P., Software Testing and Quality Assurance, New Jersey, USA, 7-10, 2008.

ÖZGEÇMİŞ

Muhammed Maruf ÖZTÜRK 1986 yılında ISPARTA'da doğdu. İlkokulu ve orta okulu Isparta'da tamamladı. Lise öğrenimini Gülkent Lisesinde tamamladı. 2008 yılında Pamukkale Üniversitesi Bilgisayar Mühendisliği bölümünden mezun oldu. 2009-2010 yılları arasında askerlik hizmetini tamamladı. 2010-2011 yılları arasında Keytorc Teknoloji firmasında yazılım test mühendisi olarak görev yaptı. 2011-2012 yıllarında Ter Yazılım firmasında web yazılımcısı olarak görev yaptı. 2011-2012 yılları arasında SDÜ Bilgisayar Mühendisliği anabilim dalında yüksek lisans eğitimi aldı. Halen, Sakarya Üniversitesi Teknoloji Fakültesi Bilgisayar Mühendisliği Bölümünde araştırma görevlisi olarak çalışmaktadır.