

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ



**KODLAMA EDİTÖRÜ GÜNLÜKLERİNİN VERİ MADENCİLİĞİ
YÖNTEMLERİYLE ANALİZİ**

LİSANS BİTİRME ÇALIŞMASI

Fatih ATEŞ

Bilgisayar Mühendisliği Bölümü

Temmuz, 2022

BURSA TEKNİK ÜNİVERSİTESİ
MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ



**KODLAMA EDİTÖRÜ GÜNLÜKLERİNİN VERİ MADENCİLİĞİ
YÖNTEMLERİYLE ANALİZİ**

LİSANS BİTİRME ÇALIŞMASI

Fatih ATEŞ
19360859074

Bilgisayar Mühendisliği Bölümü

Danışman: Prof. Dr. Turgay Tugay BİLGİN

Temmuz, 2022

BTÜ, Mühendislik ve Doğa Bilimleri Fakültesi Bilgisayar Mühendisliği Bölümü'nün 19360859074 numaralı öğrencisi Fatih ATEŞ, ilgili yönetmeliklerin belirlediği gerekli tüm şartları yerine getirdikten sonra hazırladığı “Kodlama Editörü Günlüklerinin Veri Madenciliği Yöntemleriyle Analizi” başlıklı bitirme çalışmasını aşağıda imzaları olan jüri önünde başarı ile sunmuştur.

Danışmanı : **Prof. Dr. Turgay Tugay BİLGİN**
Bursa Teknik Üniversitesi

Jüri Üyeleri : **Dr. Öğr. Üyesi Adı SOYADI**
Bursa Teknik Üniversitesi

Öğr. Gör. Dr. Adı SOYADI
Bursa Teknik Üniversitesi

Savunma Tarihi : 07 Temmuz 2022

BM Bölüm Başkanı : Prof. Dr. Turgay Tugay BİLGİN
Bursa Teknik Üniversitesi/...../.....

İNTİHAL BEYANI

Bu bitirme alışmasında grsel, işitsel ve yazılı biçimde sunulan tüm bilgi ve sonuçların akademik ve etik kurallara uyularak tarafımdan elde edildiğini, bitirme alışması içinde yer alan ancak bu alışmaya özgü olmayan tüm sonuç ve bilgileri bitirme alışmasında kaynak göstererek belgelediğimi, aksinin ortaya ıkması durumunda her türlü yasal sonucu kabul ettiğimi beyan ederim.

Öğrencinin Adı Soyadı: Fatih ATEŞ

İmzası :

ÖNSÖZ

Bu tez merhum büyükannem Birsen ENGİN'e ve her zaman yanımda olan sevgili aileme ithaf edilmiştir.

PyNar projesinin sahibi, değerli bölüm başkanım ve aynı zamanda tez danışmanım olan Prof. Dr. Turgay Tugay BİLGİN'e yol göstericiliği ve yardımları için bir teşekkürü borç bilirim.

Bu çalışmanın tez yazım aşamasında bana desteklerini esirgemeyen Beyza BAŞDEMİR'e şükranlarımı sunarım.

Temmuz 2022

Fatih ATEŞ

İÇİNDEKİLER

Sayfa

ÖNSÖZ	v
İÇİNDEKİLER	vi
KISALTMALAR	vii
SEMBOLLER	viii
ŞEKİL LİSTESİ	ix
ÖZET	x
SUMMARY	xi
1. GİRİŞ	12
1.1 Tezin Amacı	13
1.1.1 Ortalama süre analizi	13
1.1.2 Benzerlik grafi.....	14
1.1.3 Hata analizi	14
1.2 Literatür Araştırması	14
1.3 Hipotez	16
2. METHOD	17
2.1 PyNar.....	17
2.2 Veritabanı	20
2.3 PyNar Yargıç.....	23
2.3.1 Ortalama süre analizi	23
2.3.1.1 Uygulama	25
2.3.2 Benzerlik grafi analizi	27
2.3.2.1 Betik sözcüklerini ayırıştırma	28
2.3.2.2 Levenshtein mesafesi	29
2.3.2.3 Graf oluşturma	30
2.3.2.4 Uygulama	31
2.4 PyNar Hata Analizi	32
2.4.1 Uygulama	33
3. SONUÇ	35
3.1 Öneriler.....	35
KAYNAKLAR	37
ÖZGEÇMİŞ	39

KISALTMALAR

AI	: Artificial Intelligence - Yapay Zeka
COVID-19	: Koronavirüs hastalığı
DNN	: Deep Neural Networks - Derin Sinir Ağları
IP	: Internet Protocol - İnternet Protokolü
JWT	: JSON Web Token - JSON Web Jetonu
KAIST	: The Korea Advanced Institute of Science & Technology
LM	: Levenshtein Mesafesi
LSTM	: Long short-term memory - Uzun kısa süreli bellek
MERN	: MongoDB - Express.js – React.js – Node.js
NoSQL	: Not only SQL - Sadece SQL değil
PHP	: Hypertext Preprocessor
PoC	: Proof of Concept - Kavram Kanıtı
REST	: Representational State Transfer - Temsili Durum Transferi
RegEx	: Regular Expressions - Düzenli ifadeler
RNN	: Recurrant Neural Networks - Yinelemeli Sinir Ağları
SQL	: Structured Query Language - Yapısal Sorgulama Dili
SVG	: Scalable Vector Graphics - Ölçeklenebilir Vektör Grafikleri

SEMBOLLER

$t_{ogrenci}$: Öğrencinin sınav süresi
tn	: Öğrencinin n. eylemindeki cihazın zaman damgası
$t_{sınav}$: Sınav sürelerinin kümülatif ortalaması

ŞEKİL LİSTESİ

Sayfa

Şekil 2.1 : PyNar sunucusu durum diyagramı	17
Şekil 2.2 : PyNar istemci durum diyagramı	18
Şekil 2.3 : PyNar sistem mimarisi	19
Şekil 2.4 : PyNar veritabanı tablo güncellemeleri	22
Şekil 2.5 : PyNar veri tabanı Varlık-İlişki diyagramı	22
Şekil 2.6 : Üç farklı eylemin oluşturduğu günlük nesneleri	23
Şekil 2.7 : Adım 1: Öğretmen giriş ekranı	25
Şekil 2.8 : Adım 2: Sol menüdeki “Sınav” butonu	26
Şekil 2.9 : Adım 3: “bitirme_sınav” adlı sınavın “sonuçlar” butonu.....	26
Şekil 2.10 : Adım 4: Sınavı analiz et butonu	26
Şekil 2.11 : Adım 5: Ortalama süre analiz penceresi	27
Şekil 2.12 : Python ile yazılmış bir script	28
Şekil 2.13 : Jetonlarına ayrılmış Python betiğinin bir parçası	28
Şekil 2.14 : Moretokenize ile ayrıştırılmış bir Python betiğinin çıktısı	29
Şekil 2.15 : Monday ve Friday kelimelerinin LD hesaplaması	30
Şekil 2.16 : Benzerlik parametreleri, tekrar analiz butonu ve yakınlaştırma ayarları	31
Şekil 2.17 : Benzerlik grafi örneği (isimler gerçek kişilerden bağımsızdır).....	32
Şekil 2.18 : Adım 6: PyNar ortalama hata analizi	34

KODLAMA EDITÖRÜ GÜNLÜKLERİNİN VERİ MADENCİLİĞİ YÖNTEMLERİYLE ANALİZİ

ÖZET

PyNar Türkiye'nin ilk tamamen Türkçe arayüze sahip Python Kod Editörüdür. Öğrenciler PyNar Kod Editörü üzerinde geliştirme yapabilir ve geliştirdikleri betikleri PyNar Kutu özelliği sayesinde bulut ortamına yükleyebilir, diledikleri cihazdan hesaplarına erişip bulut üzerinden betiklerini indirerek geliştirmeye devam edebilirler. PyNar Kod Editörü üzerinde yapılan çalışma zamanı hataları yapay zeka tabanlı PyNar Chatbot tarafından yakalanarak öğrencilere düzeltme önerileri gönderilir. Yakalanan bu hatalar aynı zamanda sistem üzerinde günlüklere kaydedilir. PyNar öğrencilerin yanı sıra öğretmenleri de içine alan bir ekosisteme sahiptir. Öğretmenler öğrencilerine PyNar Web Portalı üzerinden ödev gönderebilir ve öğrencilerini sınav yapabilirler. Öğrenciler kendilerine atanan ödevleri yerine getirerek öğretmenlerine gönderebilir ve öğretmenler bu ödevleri yorumlayıp notlandırabilirler. Öğretmenlerin oluşturduğu sınavlar belirttikleri tarihlerde öğrencilerin aktifliğine sunulur ve öğrenciler belirlenen süre içerisinde istenenleri gerçekleştirerek betiklerini sisteme yüklerler. Sınav cevapları sisteme yüklenirken sınav esnasında oluşan günlük de sistem üzerine arkaplanda yüklenir. Ödev sisteminde olduğu gibi sınav sisteminde de notlandırma ve yorumlama sistemi mevcuttur. Bu çalışma sayesinde öğretmenlerin sınav sisteminde notlandırmanın ve yorumlamanın yanı sıra sınavın analizini de gerçekleştirebilmesine olanak sağlanmıştır. Sınav analizlerinde öğrencilerin sınavı bitirme zamanlarının ortalamasına göre sınavı ne kadar sürede bitirdiği, öğrencilerin yaptıkları hatalar ve kopya çekmiş olma ihtimali yüksek öğrencileri inceleme işlemleri gerçekleştirilebilir. Gerçekleştirilen analizler öğrencilerin gönderdiği sınav cevaplarına ve öğrencilerin gönderdiği günlük dosyaları üzerinden gerçekleştirilmektedir. Gönderilen sınav cevapları diğer öğrencilerin cevaplarıyla karşılaştırılarak öğrenciler arasındaki benzerliğe dayalı bir graf elde edilir. Elde edilen bu grafa benzerlik seviyeleri öğretmenler tarafından değiştirilebilmektedir. Graflar üzerinden direkt olarak öğrencileri incelemek öğrenci sayısı arttıkça zorlaşabilmektedir. Bunu engellemek için günlüklerden elde edilen ortalama süre analizi sayesinde incelenmesi gereken öğrenciler daha az indirgenerek öğretmenin iş yükünün düşürülmesi de amaçlanmıştır. PyNar Hata Analizi özelliği sayesinde öğrencilerin sınavlarda en çok yaptıkları hatalar analiz edilerek öğrencilerin gelişimlerine olumlu katkı sağlamak amaçlanmıştır. PyNar Yargıç ve Pynar Hata Analizi mekanizmaları öğrencilere ve öğretmenlere daha iyi bir eğitim hizmeti/deneyimi sunabilmek amacıyla geliştirilmiştir.

Anahtar kelimeler: PyNar, Kod editörü, Web Portalı, ortalama süre analizi, hata analizi, kopya analizi

ANALYSIS OF CODING EDITOR LOGS WITH DATA MINING METHODS

SUMMARY

PyNar is Turkey's first completely Turkish-language Python Code Editor. Students can develop on the PyNar Code Editor and upload the scripts they have developed to the cloud environment through the PyNar Kutu feature, they can access their accounts from any device and continue to develop by downloading their scripts from the cloud. Runtime errors made on the PyNar Code Editor are caught by the AI-based PyNar Chatbot, and correction suggestions are sent to the students. These caught errors are also logged on the system. PyNar has an ecosystem that includes teachers as well as students. Teachers can send assignments to their students through the PyNar Web Portal and they can take their students to exams. Students can fulfill their assigned assignments and send them to their teachers, and teachers can evaluate and grade these assignments. The exams created by the teachers are presented to the students' on the dates the teachers specify, and the students upload their scripts to the system by performing the requested questions within the specified time. While the exam answers are uploaded to the system, the logs created during the exam is uploaded to the system in the background. As in the assignment mechanism, there is a grading and evaluation mechanism in the exam system. Teachers can perform analysis of the exam as well as grading and evaluation in the exam system. In exam analysis, it is possible to examine how long the students took the exam according to the average of the exam completion times, the mistakes made by the students and the students who are likely to cheat. The analyzes are carried out on the exam answers sent by the students and the log files sent by the students. A graph based on the similarity between students is obtained by comparing the submitted exam answers with the answers of other students. The similarity levels to this graph can be changed by the teachers. Examining students directly through graphs can become more difficult as the number of students increases. In order to prevent this, it is aimed to reduce the workload of the teacher by reducing the students who need to be examined through the average time analysis obtained from the logs. Through the PyNar Error Analysis feature, it is aimed to contribute positively to the development of the students by analyzing the mistakes that students make the most in the exams. PyNar Yargic and Pynar Error Analysis mechanisms have been developed to provide a better educational service/experience to students and teachers.

Keywords: PyNar, Code Editor, Web Portal, average time analysis, error analysis, cheating analysis

1. GİRİŞ

PyNar, öğrencilerin kodlama deneyimini bir üst seviyeye çıkarmak, onlara özel deneyimler sunmak ve kişisel gelişimlerine katkıda bulunmak için geliştirilmiş Türkiye'nin ilk tamamen Türkçe arayüze sahip Python Kod Editörüdür. Öğrenciler PyNar Kod Editörü üzerinde geliştirme yapabilir ve geliştirdikleri betikleri PyNar Kutu özelliği sayesinde bulut ortamına yükleyebilir, diledikleri cihazdan hesaplarına erişip bulut üzerinden betiklerini indirerek geliştirmeye devam edebilirler. PyNar, öğrencilerin yaptıkları hatalara PyNar Chatbot sayesinde düzeltmeler önererek deneyimli birisinden destek alıyormuşçasına kodlarındaki hataları pozitif bir duruma çevirerek gelişimlerine katkıda bulunur.

COVID-19 dönemi öncesinde Bilgisayar tabanlı bölümlerde yüz yüze eğitimin gerekliliği yıllardır tartışma konusu olmuştur. Online eğitim için geliştirilmiş çeşitli araçlar bulunmaktadır. Bu çalışmanın geliştirilme sürecinde COVID-19 salgını ile beraber online eğitim platformlarının ne kadar hayati olduğu da gerçek yaşantımızda ortaya koyulmuştur. Nitekim online eğitimin getirdiği çeşitli sorunlar vardır. Bunlardan bazıları sınava fesat karıştırma, aldatmaya yönelik hareketler gibi durumlardır.

Bu çalışmanın vizyonu, sınav esnasında aldatmaya yönelik hareket gerçekleştirenleri tespit etmek ve kodlama esnasında çokça yapılan hataları tespit ederek öğrencilerin kişisel gelişimine destek olmaktır.

Bu çalışmanın misyonu, halihazırda Kod Editörü ve Web Portalı ile komple bir araç olan PyNar'ı bir üst seviyeye taşıyacak ve sınav sistemi açısından diğer benzer sistemlerden ayıracak önemli bir özellik olan “sınav analizi” sistemiyle PyNar'ı tamamiyle benzersiz kılmaktır.

1.1 Tezin Amacı

Servis tabanlı geliştirilen PyNar kod editörü ve Pynar Web Portal'ı sayesinde öğrenciler sınavlarını sistem üzerinden verebilmektedirler. Ancak kopya çekilmesi gibi durumlara karşı PyNar'da savunmasız kalmaktadır. PyNar modülü üzerine öğretmenler için entegre edilen PyNar Yargıç sistemiyle birlikte aşağıdakilerin sağlanabilmesi amaçlanmaktadır:

- Öğrenciler arasında ortalama bitirme süresine göre ortalamadan yüksek öğrencilerin işaretlenmesi
- Öğrencilerin gönderdikleri cevaplar üzerinden birbirleri arasında bir benzerlik matrisi kurularak bu matris üzerinde benzerlik düzeyi ayarlanabilen bir graf üretilmesi

PyNar Yargıç'ın yanı sıra öğrencilerin kişisel gelişimlerini desteklemek amacıyla en çok yapılan hatalar analiz edilerek PyNar Web Portal üzerinden sunulması amaçlanmaktadır.

PyNar üzerinde sınavların analizlerini gerçekleştirmek normal bir istemci isteğinden uzun süreler alabilmektedir ve dolaylı yoldan kaynakların dolmasına yol açabilmektedir. Geliştirilen analiz araçları PoC mantığıyla geliştirildiği için kaynak yönetimiyle ilgilenilmemiş, doğrudan ortaya çalışabilir bir sistem çıkarmaya odaklanılmıştır. PyNar'ın halihazırda bulunan RESTful servisinin içerisine inşa edilen PyNar Yargıç sistemi sayesinde öğrencilerin güvenilir bir sistem üzerinde gönül rahatlığıyla sınav olabilmesi ve öğrencilerin hatalarının analiz edilerek kazanımlarındaki eksikliklerin giderilmesini sağlanarak kişisel gelişimlerine destek olunması amaçlanmıştır.

1.1.1 Ortalama süre analizi

Sisteme gönderilen cevaplar ile birlikte sınav esnasındaki hal ve hareketlerin kaydedildiği bir günlük dosyası da sunucu üzerine yüklenmektedir. Yüklenen bu

günlük dosyasında öğrencinin sınav esnasındaki hal ve hareketleri bulunmaktadır. Günlük dosyası içerisinde öğrencinin anlık olarak gerçekleştirdiği eylemler üzerindeki zaman damgaları sayesinde sınav süresi hesaplanmaktadır. Öğrencilerin kişisel sınavı bitirme sürelerinden birikimli ortalama sınav süresinin hesaplanmaktadır. Öğretmenin analiz penceresinde ortalama süreden daha uzun sürelerde bitiren öğrenciler işaretlenerek ayrıştırılmaktadır. Öğretmen de işaretlenmiş kişiler üzerinden diğer analizleri gerçekleştirerek aldatmaya yönelik bir hareket olup olmadığını karar verebilmektedir.

1.1.2 Benzerlik grafi

Öğrencilerin gönderdikleri sınav cevapları içerisindeki kelimelerin konumlarına, uzunluklarına göre bir eşleştirme sistemi sayesinde her öğrencinin birbiriyle olan benzerlik skorları bir matris üzerinde üretilmektedir. Üretilen bu matris üzerinden öğretmenin belirleyebildiği benzerlik oranında bir benzerlik grafi ortaya çıkarılmaktadır. Öğretmen PyNar Yargıç üzerinden bu analize erişebilir, benzerlik oranını güncelleyebilir, ortalama süre analizinde anomali tespit edilen öğrenciler üzerinde incelemeler gerçekleştirebilmektedir. Oluşturulan bu graf sayesinde aldatmaya yönelik hareketlerin ortaya çıkarılabilmesi amaçlanmıştır.

1.1.3 Hata analizi

Öğrenciler sınavlarını sunucu üzerine yüklerken yanlarında yüklenen günlükler içerisindeki eylemlerde yaptıkları hatalar da bulunmaktadır. Öğrencilerin ortak olarak yaptıkları hatalar veri tabanı üzerinde kayıtlanarak öğretmenin analizine sunulmaktadır. Bu hata analizi öğrencilerin kendi gelişimlerini iyileştirmek için kullanılabilecektir.

1.2 Literatür Araştırması

Tiong and Lee (2021), özellikle COVID-19 salgınında sırasında artan çevrimiçi sınavlarda kopya çekmenin önüne geçmeyi amaçlamışlardır. Bu kapsamda, internet protokolü ağ dedektörü ve derin öğrenme tabanlı davranış algılama dedektörü geliştirerek bunun üzerinden bazı yapay zeka tekniklerini önermişlerdir. Önerilen yöntem ile öğrencilerin hareketleri kontrol edilerek olası kopyayı tespit edebilme ve engelleyebilme kabiliyetine sahip olduğunu savunmuşlardır. Yöntemin etkinliğini

gösterebilmek için ara sınav ve final sınavları veri setleri kullanılmış ve DNN, DenseLSTM, LSTM ve RNN adlı dört derin öğrenme algoritması üzerinde uygulama gerçekleştirilmiştir. Sonuçlar, DNN için %68, DenseLSTM için %95, LSTM için %92 ve RNN için %86'lık doğruluk oranı vermiştir. Bu doğruluk oranları doğrultusunda sınav sonuçlarının tekrardan incelenmesi için yeterli uyarı olarak görülmektedir.

Byun et al. (2020), çevrimiçi gerçekleştirilen programlama derslerinde gerçekleşen kopyaların önüne geçmeyi amaçlamışlardır. Bu kapsamda, programlama dersi alan sınıflarda yapılan alıştırmalarda tuş vuruş dinamikleri yaklaşımı sunmuşlardır. Bu yaklaşımla, yapılan alıştırmayı gerçekleştiren kişilerin klavye ile ilgili hareketleri incelenerek kaydedilmekte ve analiz edilerek olası kopya durumunda öğretim elemanı ve personel bilgilendirilmektedir. Yaklaşımın değerlendirilmesi için KAIST'te Programlamaya Giriş dersinde 27 öğrenciye test yapılmıştır. Sonuçlar, önerilen yaklaşımın %94 doğruluk oranıyla kopya tespiti yapabildiğini göstermektedir.

Kustanto and Liem (2009), programlama sınıflarında kısa kaynak kodlarında meydana gelen kopyaları engellemeyi amaçlamışlardır. Bu kapsamda, iki adımda çalışan Deimos'u tasarlamış ve geliştirmişlerdir. İlk olarak kaynak kod parçalara bölünmüş ve daha sonra jetonlara dönüştürülmüştür. Daha sonra Running Karp Rabin Greedy String Tiling algoritmasını kullanılarak jeton hâlindeki parçalar karşılaştırılmıştır.

Obadi et al. (2010), bir veri madenciliği yöntemi olan spektral kümeleme ile online eğitimdeki öğrenci gruplarının davranış kalıplarının akademik performansı bir başka deyişle notları arasındaki ilişkiyi bulmayı amaçlamışlardır. Bu kapsamda, spektral kümeleme yöntemi ve grafik üretimindeki girdilerin ve boyutların ayarlanması için yazılım geliştirmişlerdir. Geliştirilen yazılım, 307 öğrenci ve benzerlik düzeyi = 0,5 olan Mikro Ekonomi adlı ders üzerinde incelenmiştir. Sonuçlar, öğrenci gruplarının davranış kalıpları ile akademik performansı arasında ilişkiye rastlanmadığını göstermektedir.

1.3 Hipotez

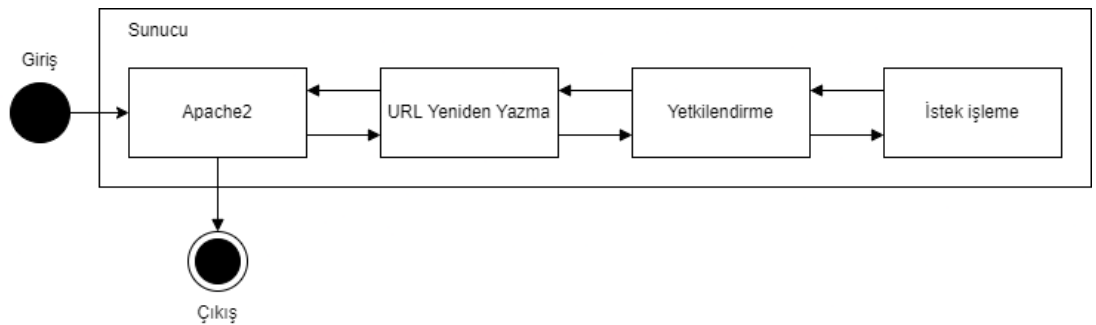
Sınavlardaki aldatmaya yönelik hareketler okulların, üniversitelerin vb. eğitim kurumlarının yıllardır çözemediği bir sorun olarak süre gelmektedir. COVID-19 salgını ile online eğitim sistemleri kullanılmaya başlanmıştır. Online eğitim sistemlerinin kullanılması sınavlar esnasında aldatmaya yönelik hareketlerin artmasına sebep olmuştur. Online ortamlarda öğrencileri gözetlemek çok daha kolay olmasına rağmen etik kurallar çerçevesinde veri toplama işlemi gerektirir. Öğrencilerin izinleri etrafında toplanan verilerin ışığında: PyNar Yargıç ve PyNar Hata Analizi, öğrencilere PyNar aracılığıyla sunulmuş olan özel bir geliştirme ortamına daha adil ve öğrenci gelişimini bir üst seviyeye taşıyacak bir çözüm getirmeyi amaçlamaktadır.

2. METHOD

2.1 PyNar

PyNar Türkiye'nin ilk tamamen Türkçe ara yüze sahip Python kod editörüdür. Öğrenciler PyNar Kod Editörü üzerinde geliştirme yapabilir ve geliştirdikleri betikleri PyNar Kutu özelliği sayesinde bulut ortamına yükleyebilir dilerseler farklı bilgisayarlardan hesaplarına erişerek bulut üzerinden betiklerini indirerek geliştirmeye devam edebilirler. PyNar üzerinde yapılan hatalar yapay zeka tabanlı PyNar Chatbot tarafından yakalanarak öğrencilere düzeltme önerileri gönderilir. PyNar öğrencilerin yanı sıra öğretmenleri de içine alan bir ekosisteme sahiptir. Öğretmenler öğrencilerine PyNar üzerinden ödev gönderebilir ve öğrencilerini sınav yapabilirler. Öğrenciler kendilerine atanan ödevleri yerine getirerek öğretmenlerine gönderebilir ve öğretmenler bu ödevleri yorumlayıp notlandırabilirler. Öğretmenlerin oluşturduğu sınavlar belirttikleri tarihlerde öğrencilerin aktifliğine sunulur ve öğrenciler belirlenen süre içerisinde istenenleri gerçekleştirerek betiklerini sisteme yüklerler. Ödev sisteminde olduğu gibi sınav sisteminde de notlandırma ve yorumlama sistemi mevcuttur. Öğretmenler sınav sisteminde notlandırmanın ve yorumlamanın yanı sıra sınavın analizini de gerçekleştirebilirler. Sınav analizlerinde öğrencilerin sınavı bitirme zamanlarının ortalamasına göre ne kadar sürede bitirdiği, öğrencilerin yaptıkları hatalar ve kopya çekmiş olma ihtimali yüksek öğrencileri inceleme işlemi gerçekleştirilebilir. Tüm özellikleriyle PyNar komple bir editördür.

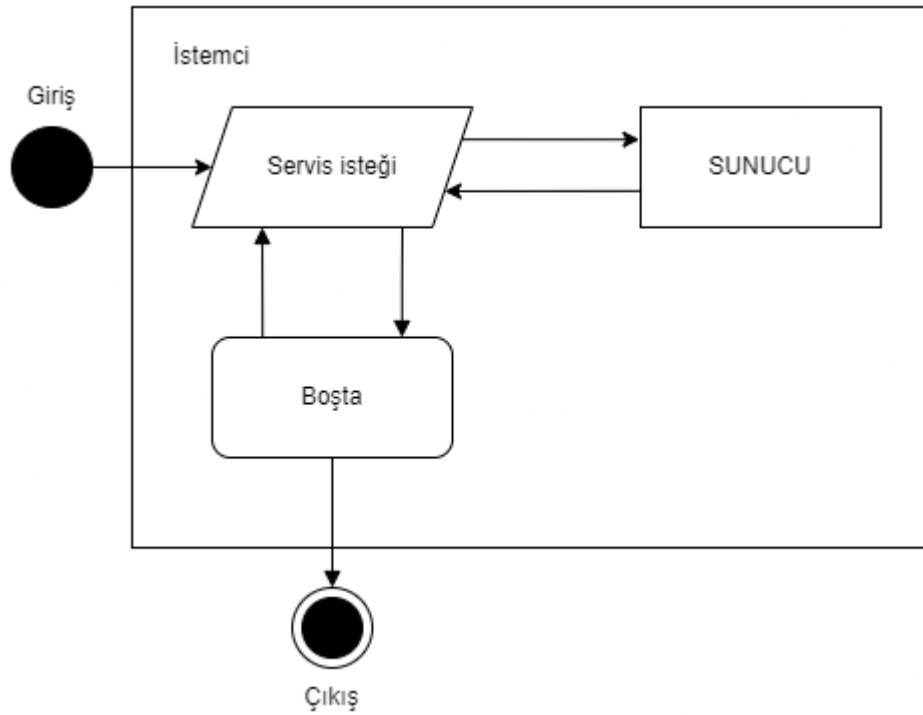
PyNar sistemi Linux tabanlı sunucularda Apache2 üzerinde koşturulacak şekilde tasarlanmıştır. PyNar sisteminin koşturulduğu bir sunucunun durum diyagramı Şekil 2.1 üzerinde verilmiştir.



Şekil 2.1 : PyNar sunucusu durum diyagramı

Sunucuya ulaşan istemci istekleri öncelikle Apache2 tarafından karşılanır. Apache2 içerisinde ilk olarak URL yeniden yazma işlemi gerçekleştirilir ve gönderilmiş olan günlük dile yakın URL sunucu üzerinde sorgu parametrelerine ayrıştırılarak tekrardan işlenir. Daha sonrasında yetkilendirme gerekli olan bir uç noktaya istek atılıyorsa yetkilendirme işlemi gerçekleştirilir. Eğer ki herkese açık bir uç noktaya istek atılıyorsa bu adım es geçilmektedir. Yetkilendirme işlemi de tamamlandıktan sonra istek artık PHP betikleri tarafından yorumlanmaya hazır bir hale getirilmektedir. PHP betikleri gerekli aksiyonları yerine getirdikten sonra geriye bir yanıt hazırlar ve içerisinde gerekli mesajları barındırarak sistemden çıkarak istemciye geri döndürmektedir.

PyNar, servis tabanlı çalışan bir sistemdir. Sunucu içerisinde bir ön yüz bir de arka yüz uygulaması bulunmaktadır. PyNar toplamda ise iki ön yüz uygulamasına sahiptir; PyNar Kod Editörü ve Pynar Web Portalı. Bu iki ön yüzden birini kullanan bir istemcinin durum diyagramı Şekil 2.2 ile verilmiştir.

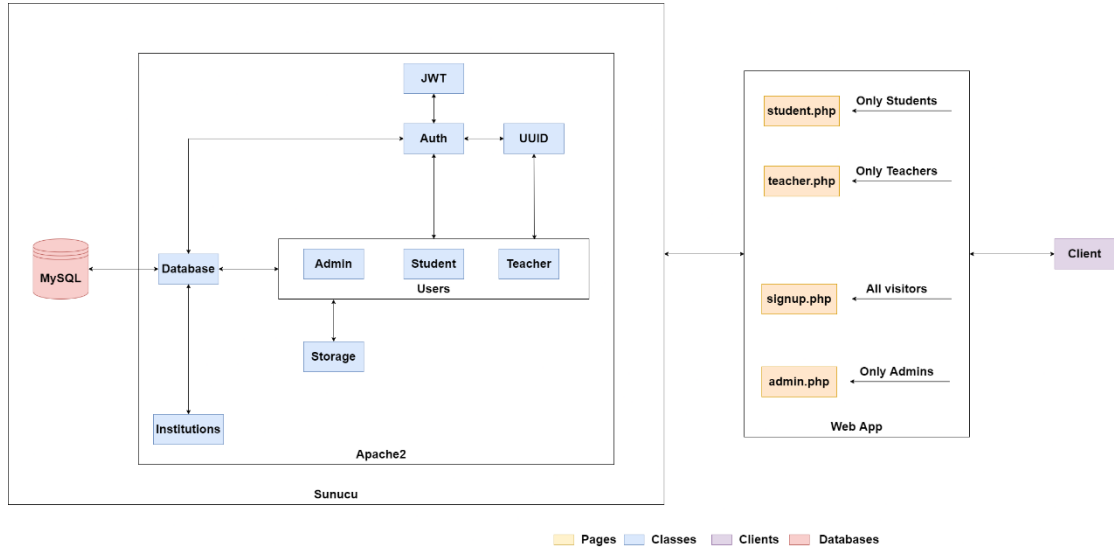


Şekil 2.2 : PyNar istemci durum diyagramı

Bir istemci ön yüz uygulamaları üzerinden temel olarak servis ile haberleşme işlemleri gerçekleştirmektedir. İstemci ön yüz uygulamasını açtığı andan itibaren işlem yapmadığı her an boşta durmaktadır. Bir işlem gerçekleştirdiğinde boşta işleminden bir servis isteği oluşturulur ve sunucu ile haberleşilir. Sunucu diyagramındaki döngü

tamamlandıktan sonra ön yüz uygulamasına bir yanıt döner ve sistem yine boşta durumuna geçer. Ön yüz uygulaması kapatıldığı andan itibaren durum diyagramından da görülebileceği üzere çıkış ile durum sonlandırılır.

PyNar, Pynar Web Portal'ı ve arka yüz servisi ile komple bir bütündür. Tüm sistemin mimarisi Şekil 2.3 üzerinde verilmiştir.



Şekil 2.3 : PyNar sistem mimarisi

Sunucu tarafında gelen bir isteğin durum diyagramı Şekil 2.1 üzerinde verilmiştir. Durum diyagramı içerisinde bulunan yetkilendirme ve istek işleme bölümleri Şekil 2.3 üzerinde açıkça görülmektedir. Yetkilendirme için JWT metodolojisi tercih edilmektedir. Oturum açmış olan kullanıcı kendi varlığına özel olan bölgelere de erişim sağlayabilmektedir. Her varlık sınıflar vasıtasıyla birbirinden ayrılmıştır ve sistemde üç adet farklı rol bulunmaktadır. Üç ayrı rolün ortak gerçekleştirebildiği ve kendi özel ayrı aksiyonları bulunmaktadır.

Şekil 2.3 üzerinde sunucudaki arka yüzü tamamlayan bir ön yüz görülmektedir. Görülen ön yüz Pynar Web Portalıdır ve bu uygulama sunucudaki bir takım aksiyonları gerçekleştirebilmesine olanak tanımaktadır. PyNar Kod Editörü ve PyNar Web Portalı ile beraber ve senkronize bir şekilde arka yüz ile haberleşerek onu tamamlamaktadır. PyNar Web Portalı öğretmen ve öğrenci için iki ayrı senaryoya sahip olmasına rağmen daha çok öğretmene hitap eden bir uygulama olarak tasarlanmıştır. Öğrenciler bir çok işlemini PyNar Kod Editörü üzerinden gerçekleştirebilmektedir. Ancak kayıt olma,

buluttaki dosyalarına erişme, ödevlerindeki notlandırmayı ve yorumlamayı görme gibi PyNar Kod Editörüne ihtiyaç duyulmayan ve herhangi bir cihazdan yapılabilecek olan işlemleri Pynar Web Portalı üzerinden gerçekleştirebilmektedirler.

Öğretmen, PyNar Kod Editörü üzerinde bir işlem hakkına sahip değildir. Yalnızca Pynar Web Portalı üzerinden işlemler gerçekleştirebilir. PyNar Kod Editörü üzerinden sınav oluşturma, sınav analizi, ödev oluşturma, ödev ve sınav notlandırma/yorumlama gibi bir çok aksiyonu yerine getirebilmektedir.

Admin, şeffaflık açısından tüm yetkilerle donatılmamıştır. Öğretmen ve öğrencinin bilgilerini yönetebilir/değiştirebilir ancak öğrencilerin sınavlarına müdahale etme gibi bir şansı bulunmamaktadır. Genel olarak yönetim amaçlı tasarlanmış bir roldür.

2.2 Veritabanı

Günümüzde verileri depolamak için kullanılan bir çok veri tabanı mimarisi bulunmaktadır. Bunlar arasında en yaygın kullanılan veri tabanı mimarileri ilişkisel veritabanları ve NoSQL veri tabanlarıdır. Bunların yanında başka veri tabanı mimarileri de olsa da ilişkisel veri tabanları geçmişten günümüze her daim çoğunluk olarak tercih edilen veritabanı mimarileri olmuşlardır. Son yıllarda giderek artan bir popülerliğe sahip olan NoSQL veri tabanları ilişkisel veritabanlarının tahtını sarsmakla kalmamış özellikle eksik verilerin bulunabildiği yapılarda vazgeçilmez bir veri tabanı modeli haline gelmiştir.

İlişkisel veri tabanlarında oluşturulmuş olan varlıklar belirli kolonlar aracılığıyla birbirleriyle ilişkilendirilmektedir. İlişkisel model, varlıkları tablolar aracılığıyla ifade edilmesine olanak sağlamaktadır. Bu tez içerisinde ilişkisel veri tabanı olarak MySQL tercih edilmiştir. MySQL Oracle şirketi tarafından geliştirilen açık kaynak kodlu bir veri tabanı yazılımıdır. Orta ölçekli projeler için oldukça tercih edilmektedir. PyNar açık kaynak kodlu bir yazılımıdır ve Pardus üzerinde kurulu gelmesi planlanmaktadır. Bu demektir ki asıl hedefi Linux kullanıcılarıdır. Bundan dolayıdır ki MSSQL veya diğer kapalı kaynak ve maliyet gerektiren veri tabanları bu yüzden tercih edilmemiştir. Açık kaynak veri tabanları arasında en güvenilir veri tabanları arasında olan MySQL veri tabanı orta ölçekli projeler için ideal bir kapı halindedir.

NoSQL, “sadece SQL değil” manasına gelmektedir. Ancak NoSQL veritabanlarının yazılım geliştirme kitleri içerisinde genelde SQL yazılarak da sorgulama gerçekleştirilebilmektedir. NoSQL veritabanları son 10 yılda veri işlemenin öneminin büyük oranda artmasıyla beraber başta büyük şirketler olmak üzere herkesin gözde veri tabanları haline gelmişlerdir. Bu çalışma içerisinde NoSQL veri tabanları arasında en çok öne çıkan döküman tabanlı MongoDB veri tabanı ile karşılaştırma gerçekleştirilmiştir. MongoDB kaynak kodu açık ve ücretli/ücretsiz sürümleri bulunan bir veri tabanıdır. Son zamanlarda MERN yapılarında çokça tercih edilmesi sayesinde bir çok yerde adını duyurmayı ve kendi kendini pazarlamayı başarmıştır.

Bu çalışma için bir veri tabanı tercihi yapılması gerekmektedir. Veri tabanının sahip olması gereken özellikler aşağıdaki gibidir:

- Hızlı olması gerekmektedir
- Kolay yönetilebilmesi gerekmektedir
- PHP desteği olmalıdır
- Açık kaynak kodlu olmalıdır
- Debian tabanlı sistemlerde çalışabilmesi gerekmektedir




Başlıca yukarıdaki özelliklere uyan iki adet veri tabanı seçilmiştir. MySQL ve MongoDB veri tabanları istenen tüm özellikleri karşılamaktadır. MySQL’in PHP ile uyumluluğu uzun zamandan beri bilinmekte ve daha güvenilir bir yapı kurulmasına olanak sağlamaktadır. PyNar sisteminde yazma ve güncellemeden çok okuma sorguları ön plana çıkacaktır dolayısıyla okuma hızı PyNar için çok daha kritik bir rol oynamaktadır. MongoDB’nin kendi resmi internet uygulamasında belirttiği gibi MySQL yüksek sayılı okuma sorgularında çok daha hızlı bir performans ortaya koymaktadır. MySQL ve PHP uyumu da göz önüne alındığından PyNar sistemi için MySQL veri tabanı tercih edilmiştir.

PyNar Yargıç ve PyNar Hata Analizi mekanizmalarını var olan PyNar sistemine entegre edebilmek için Şekil 2.4 üzerinde görüldüğü gibi iki adet yeni tablo ve bir adet yeni kolon eklenmiştir.

EXAMS_COMMON_ERRORS	
PK1,FK1	<u>exam_id</u>
PK2	<u>error</u>
	times
	created_date

EXAMS_ANALYZE	
PK,FK1	<u>exam_id</u>
	analyzed
	levenshtein_min
	levenshtein_max
	needleman_min
	needleman_max
	distance
	weight
	avg_times
	created_date

STUDENTS_BIND_EXAMS	
PK1	<u>student_id</u>
PK2,FK1	<u>exam_id</u>
UK1,FK2	<u>answer_file_id</u>
	started_at
	comment
	grade
	finished_at
	special_permission
	avg_times

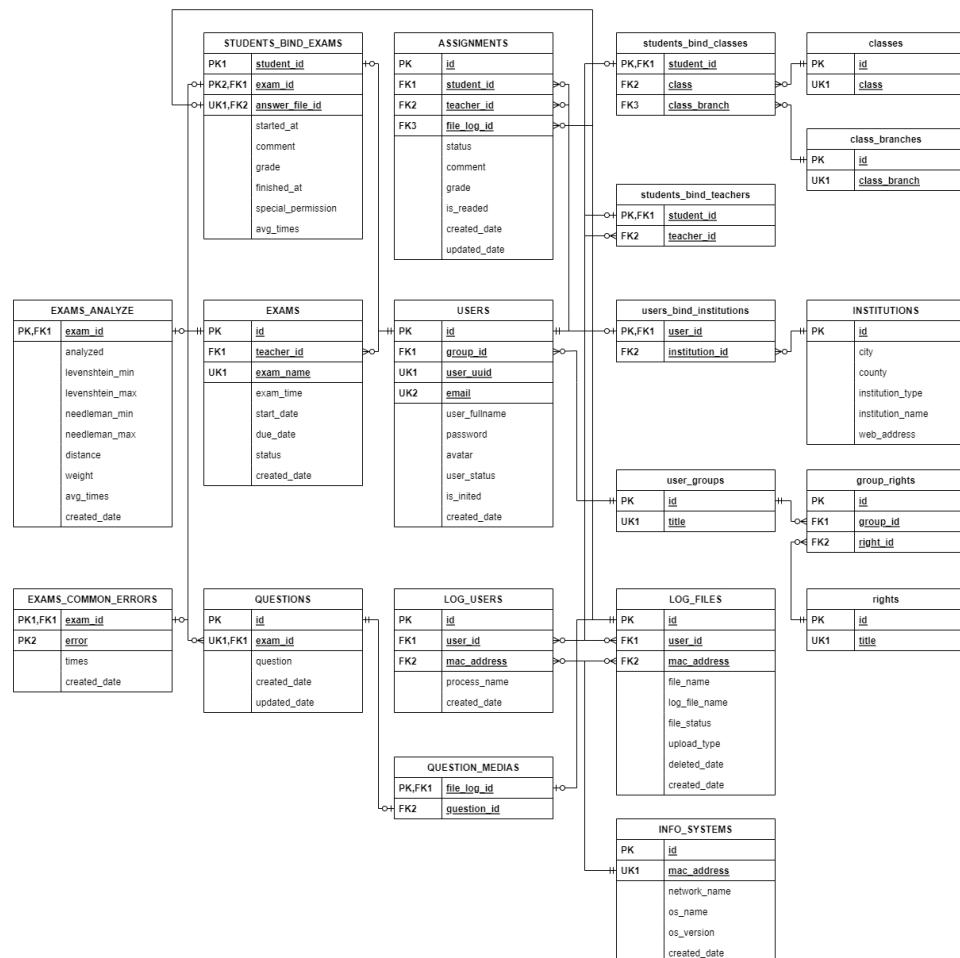
 Yeni  Güncellenmiş  Eklenen kolon

 Güncellenmiş

 Eklenen kolon

Şekil 2.4 : PyNar veritabanı tablo güncellemeleri

PyNar sistemine ait veri tabanı modelinin güncellenmiş Varlık-İlişki diyagramı Şekil 2.5 ile verilmiştir.



Şekil 2.5 : PyNar veri tabanı Varlık-İlişki diyagramı

Şekil 2.5 üzerinde görüldüğü üzere toplamda 20 adet birbirleriyle çeşitli ilişkiler içerisinde bulunan tablo bulunmaktadır.

2.3 PyNar Yargıç

Öğretmenler, PyNar sistemi üzerinden öğrencilerinin sınav vermelerini isteyebilirler. Ancak bu bir takım etik sorunların da gündeme gelmesine çanak tutmaktadır. PyNar Yargıç, online ortamlarda yapılan sınavlarda öğrencilerin aldatmaya yönelik hareketler gerçekleştirmesini reaktif bir yaklaşım ile tespit etmeyi amaçlayan bir PyNar’ın vazgeçilmez ve ayırıcı bir özelliğidir.

PyNar sistemi üzerinde öğretmen rolüne konuşturılmış ve varsayılan olarak gelen PyNar Yargıç özelliği bir sınav için gönderilmiş olan sınav cevap dosyası ve sınav günlük dosyalarını analiz eder. PyNar Yargıç sistemi iki adımlı çalışmaktadır: ortalama süre analizi, benzerlik grafi analizi.

2.3.1 Ortalama süre analizi

Ortalama süre analizi hesaplanırken, tüm öğrencilerin kendi sınavı çözme zamanlarını hesaplanır ve daha sonrasında kümülatif olarak sınav çözülme süresini hesaplanır. Şekil 2.6 ile bir günlük dosyasındaki üç adet eylemin oluşturduğu nesneler verilmiştir.

```
[
  {
    "MachineId": "866167688112",
    "time": "2022-02-15T15:13:53.157", // t1
    "action": "Basılan Tuş: r",
    "totalchars": 2,
    "totallines": 1
  },
  {
    "MachineId": "866167688112",
    "time": "2022-02-15T15:13:53.433", // t2
    "action": "Basılan Tuş: i",
    "totalchars": 3,
    "totallines": 1
  },
  {
    "MachineId": "866167688112",
    "time": "2022-02-15T15:13:53.810", // t3
    "action": "Basılan Tuş: n",
    "totalchars": 4,
    "totallines": 1
  },
]
```

Şekil 2.6 : Üç farklı eylemin oluşturduğu günlük nesneleri

Şekil 2.6 üzerinde görüldüğü üzere her nesnenin birer “time” adında özniteliği bulunmaktadır. Öğrencinin kendi sınavı tamamlama süresi bu özniteliklere dayanarak hesaplanmaktadır. Süre hesaplama formülü Denklem 2.1 ile verilmiştir.

$$t_{ögrenci} = \frac{(t3 - t2) + (t2 - t1)}{n - 1} \quad (2.1)$$

Denklem 2.1 üzerinde görüldüğü üzere zamanlar ardışık olarak birbirinden çıkarılır ve toplam nesne sayısının 1 eksiğine bölünerek öğrencinin kişisel sınav süresi mikro saniye cinsinden hesaplanır. Hesaplanan süreler veritabanı üzerinde öğrencilerin sınav ile bağlı olduğu “students_bind_exams” tablosunda “avg_times” kolonuna kaydedilir. Bir sınav cevap dosyası, bir log dosyası bulunmayan/boş olan veya sınava girmeyen öğrencilerin “avg_times” kolonu “0” olarak varsayılan değerde işaretlenmektedir.

Öğrencilerin kişisel süreleri hesaplandıktan sonra sınavın kümülatif ortalama süresi hesaplanmaktadır. Denklem 2.2 üzerinde ortalama hesaplama formülü verilmiştir.

$$t_{sınav} = \frac{\sum_{i=0}^n t_{ögrenci}}{n} \quad (2.2)$$

Denklem 2.2 üzerinde görüldüğü üzere sınava girmiş tüm öğrencilerin kendi süreleri toplanarak öğrenci sayısına bölünür ve böylece sınavın kümülatif ortalama çözülme süresi elde edilir. Elde edilen ondalık değer veritabanı üzerinde “exams” tablosunun “avg_times” adlı kolonuna kaydedilir. Bu analiz verileri her sınav için bir kere hesaplanmaktadır.

Denklem 2.2 için öğrencilerin sınava girmiş ve bir çözüm göndermiş olması kritik değerdir. Aksi takdirde bu öğrenciler sınavın kümülatif ortalama süresinin hesaplanmasında gürültü değerler olarak yer almış olmaktadır. Gürültü değerler ise analizin doğruluğunu istemeden de olsa yoldan saptırabilecek değerlerdir. Eğer ki milyonlarca öğrencinin girdiği bir sınav ise gürültü değerler toplam kütleyle göre oldukça az kalacağından dolayı dahil edilmeleri büyük sorun teşkil etmeyecektir. Ancak bir sınıfta ortalama 80 kişi olacağı düşünüldüğünde bu değerler analizi gereğinden fazla saptıracaktır. Dolayısıyla bu değerler dahil edilmemiştir.

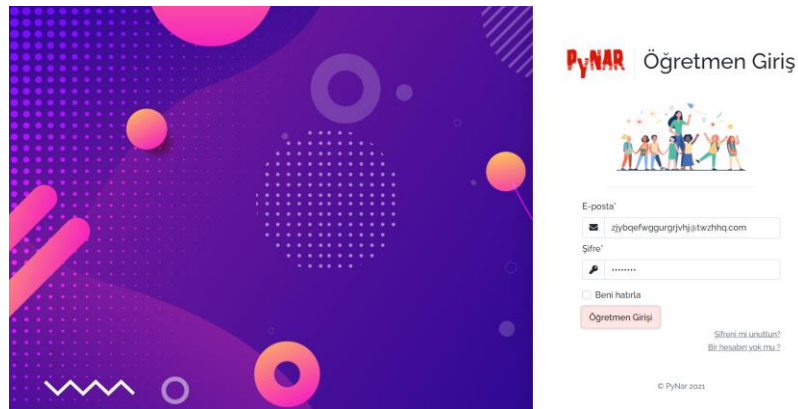
Hesaplanan veriler Pynar Web Portal'ı üzerinden öğretmenin kullanımına sunulur. Sunulan veriler üzerinde ortalamanın üzerinde bir sınav çözme süresine sahip olan öğrenci varsa bu öğrenciler kırmızı renk ile işaretlenir. Kırmızı olarak işaretlenen öğrencilerin günlük dosyalarının, kod büyüme grafiklerinin, sınav cevap dosyalarının ve benzerlik grafi analizindeki konumlarının incelenmesi tavsiye edilmektedir.

2.3.1.1 Uygulama

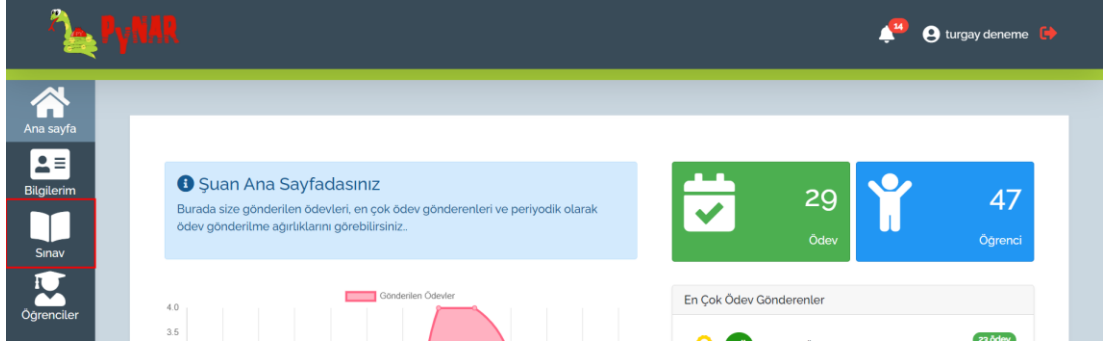
PyNar sistemi üzerinde süre analizini Pynar Web Portal'ı üzerinden gerçekleştirebilmek için sırasıyla aşağıdaki adımlar uygulanmalıdır:

- Adım 1: Öğretmen hesabına giriş yapılır
- Adım 2: Sol menüden “Sınav” sekmesine tıklanır
- Adım 3: Sınav sekmesinde analiz yapılmak istenen sınav satırı üzerinde “sonuçlar” butonuna tıklanır
- Adım 4: Açılan pencerede “Sınavı Analiz Et” adlı butona tıklanır
- Adım 5: Yönlendirilen sekme içerisinde bir tablo üzerinde öğrencilerin sınav süreleri bulunmaktadır

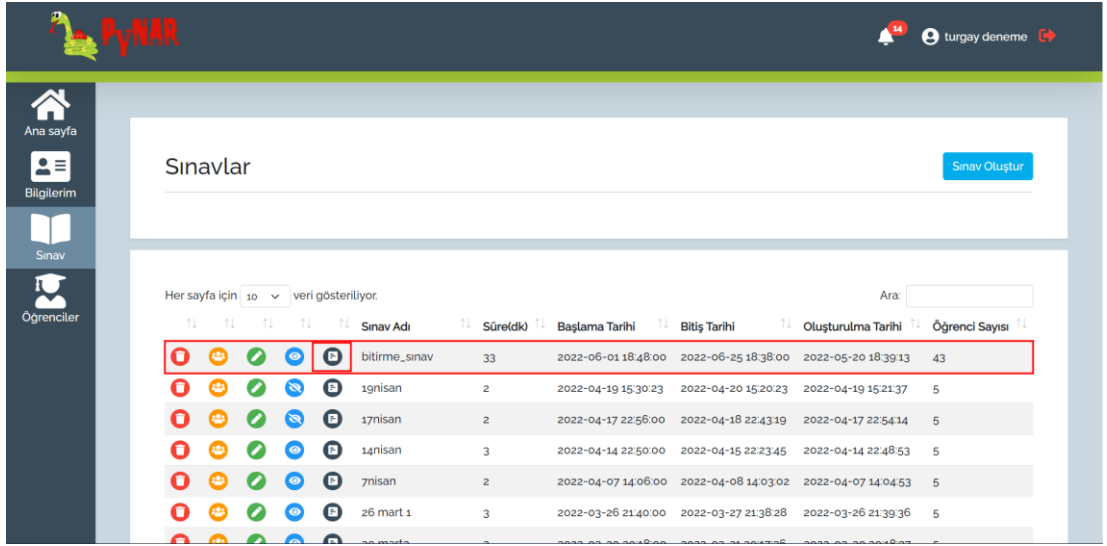
Yukarıdaki adımlar sırasıyla tamamlandıktan sonra açılmış olan en son pencere içerisindeki tabloda kırmızı işaretlenmiş öğrenciler incelenmesi önerilen öğrencilerdir. Aynı pencere içerisinde tablonun üst kısmında “ortalama sınav süresi” belirtilmiştir. Yine aynı pencere üzerinden “PyNar Hata Analizi” ve “Benzerlik Grafi Analizi” butonlarına tıklanarak diğer analizler de gerçekleştirilebilmektedir. Aşağıdaki şekiller ile uygulamalı olarak gösterilmiştir.



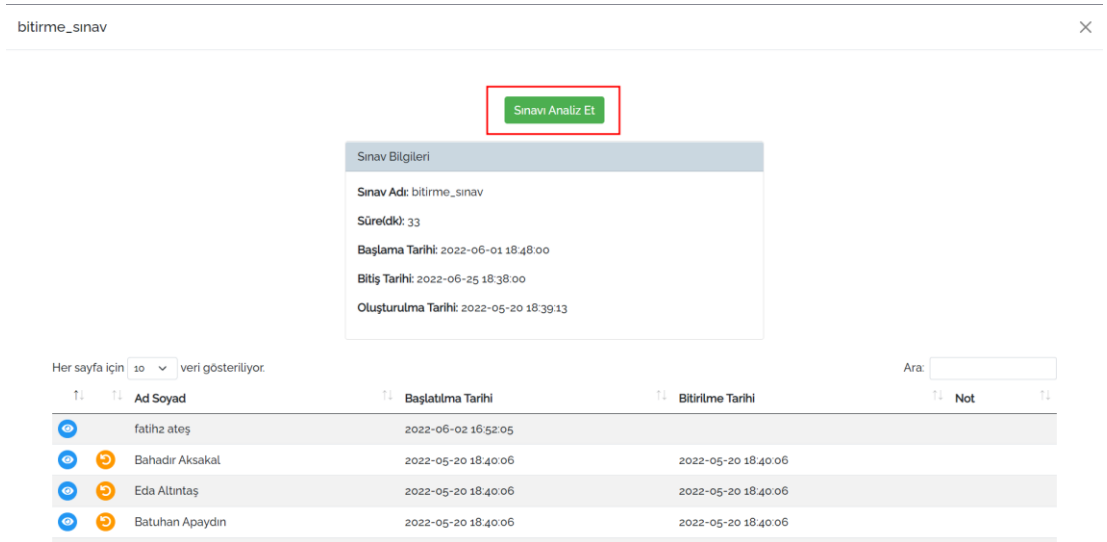
Şekil 2.7 : Adım 1: Öğretmen giriş ekranı



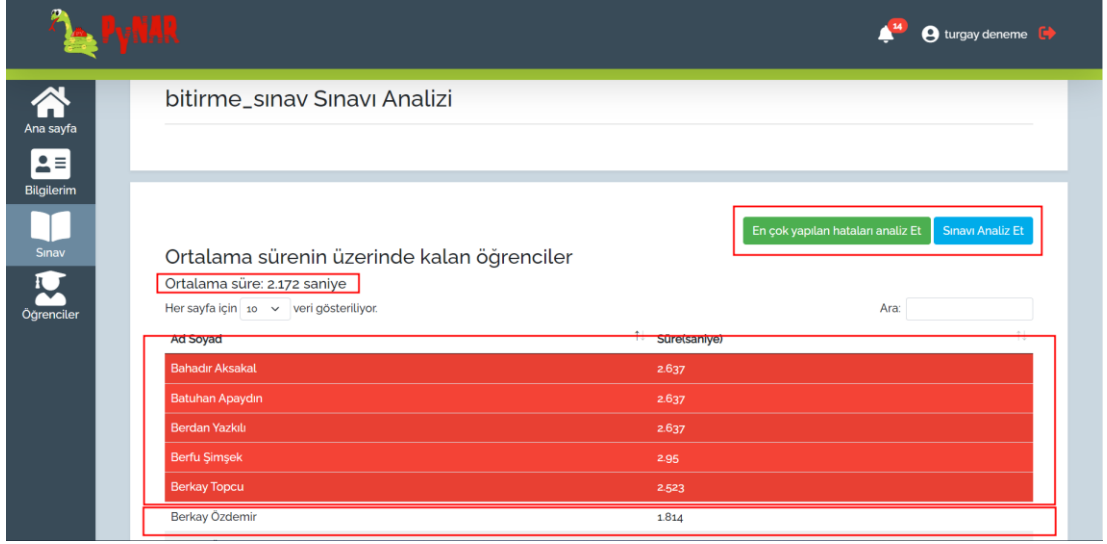
Şekil 2.8 : Adım 2: Sol menüdeki “Sınav” butonu



Şekil 2.9 : Adım 3: “bitirme_sınav” adlı sınavın “sonuçlar” butonu



Şekil 2.10 : Adım 4: Sınavı analiz et butonu



Şekil 2.11 : Adım 5: Ortalama süre analiz penceresi

2.3.2 Benzerlik grafi analizi

Benzerlik grafi analizi, temelde öğrencilerin gönderdikleri sınav cevap dosyalarına bağlıdır. Öğrencilerin gönderdikleri sınav cevap dosyaları öncelikle jetonlarına ayrılır. Ayrıştırma gerçekleştirildikten sonra öğrencilerin cevapları arasında LM hesaplama yöntemine göre Levenshtein skoru üretilir. Üretilen skorlar bir matris içerisinde birleştirilirler. Toplamda N burada öğrenci sayısı olmak üzere $N*N/2$ skor hesaplanır. LM değeri ne kadar küçük ise o öğrencilerin cevap dosyalarının yapıları birbirine o kadar benziyor anlamına gelmektedir. Elde edilen matris sayesinde öğrencilerin birbirlerine olan benzerliklerinden yola çıkılarak NetworkXX kütüphanesi kullanılarak SVG formatında bir çıktı üretilir. Üretilen bu çıktı PyNar dosya saklama sistemi içerisinde sınav dosyalarının kaydedildiği “exam_media” adlı klasör altında öğretmenin tekil kimliği adına açılan dosya içerisinde “sınav_adı_lev.svg” isim formatında kaydedilir. Kaydedilen dosya SVG formatında olduğu için internet uygulamalarında sorunsuz bir şekilde gösterilebilmektedir. Ancak burada bir benzerlik oranı parametresi bulunmaktadır. Bu benzerlik parametresi kod uzunluğuna, kod içerisindeki sözcük sayısına vb. parametrelere dayalı olarak değiştirilmesi gerekmektedir. Bu çalışma içerisinde bunun için bir yaklaşım geliştirilmemiş bu ayarı yapma şansı öğretmenin inisiyatifine bırakılmıştır. Varsayılan olarak üretilen benzerlik grafinin benzerlik değeri Pynar Web Portal’ı üzerinden değişiklik gerçekleştirilebilmektedir.

2.3.2.1 Betik sözcüklerini ayrıştırma

Jetonlarına ayırma işlemi için Python çalıştırılabilir dosyasının “tokenize” aracı kullanılarak gerçekleştirilir. Şekil 2.12 üzerinde bir sınav cevap dosyası ve Şekil 2.13 üzerinde Şekil 2.12 üzerindeki dosyanın jetonlaştırılmış hali verilmiştir.

```
def tokenize_current_keywords(print_function=False):
    if print_function is True:
        return [x for x in KEYWORDS if x != "print"]
    else:
        return KEYWORDS

def tokenize_generator(sequence, print_function=False):
    current_keywords = tokenize_current_keywords()
    for item in sequence:
        if item in current_keywords:
            yield [item.upper(), item]
            continue

        for candidate, token_name in TOKENS:
            if candidate.match(item):
                yield [token_name, item]
                break
        else:
            raise UnknowItem("Can't find a matching token for this item: '%s'"
                              % item)
    yield ('ENDMARKER', '')
    yield
```

Şekil 2.12 : Python ile yazılmış bir script

```
> python3 -m tokenize demo.py | cat - -n
1 0,0-0,0:      ENCODING      'utf-8'
2 1,0-1,3:      NAME          'def'
3 1,4-1,29:     NAME          'tokenize_current_keywords'
4 1,29-1,30:    OP           '('
5 1,30-1,44:    NAME          'print_function'
6 1,44-1,45:    OP           '='
7 1,45-1,50:    NAME          'False'
8 1,50-1,51:    OP           ')'
9 1,51-1,52:    OP           ':'
10 1,52-1,53:   NEWLINE     '\n'
11 2,0-2,4:     INDENT       '    '
12 2,4-2,6:     NAME          'if'
13 2,7-2,21:    NAME          'print_function'
14 2,22-2,24:   NAME          'is'
15 2,25-2,29:   NAME          'True'
16 2,29-2,30:   OP           ':'
17 2,30-2,31:   NEWLINE     '\n'
18 3,0-3,8:     INDENT       '        '
19 3,8-3,14:    NAME          'return'
20 3,15-3,16:   OP           '['
21 3,16-3,17:   NAME          'x'
22 3,18-3,21:   NAME          'for'
```

Şekil 2.13 : Jetonlarına ayrılmış Python betiğinin bir parçası

Şekil 2.13 üzerinde görüldüğü gibi 4 ayrı kolon vardır ve analiz için yalnızca 3. kolon kullanılmaktadır. Tokenize aracı ile bir betik ayrıştırıldığında 2. satırdaki “def” sözcüğünü bir isimlendirme olarak görmesine rağmen aslında bu sözcük bir öntanımlı ifadedir. Bu şekilde bir ayrıştırma işlemi benzerlik grafinin çizilmesinde yanlış etkilere sebebiyet vermektedir. Dolayısıyla burada tokenize aracı direkt olarak kullanılmamaktadır. Bir geliştirilmiş versiyonu olan “moretokenize” aracı ile ayrıştırma gerçekleştirilmektedir.

Moretokenize, tokenize aracının özelliklerinin üstüne Python’ın öntanımlı ifadelerini de dahil ederek çok daha geniş bir portföy sunmaktadır. Moretokenize, önceden tanımlamış olduğumuz RegEx kuralları ve öntanımlı ifadelerin “KEYWORDS” adında bir dizi ile verilmesiyle ayrıştırma işlemini çok daha kararlı ve kullanılabilir bir biçimde gerçekleştirmektedir. Şekil 2.14 üzerinde moretokenize ile aynı Python betiğinin ayrıştırıldığında nasıl bir çıktı elde edildiği verilmiştir.

```
> python3 test.py
['DEF', 'VAR', 'LEFT_PARENTHESIS', 'VAR', 'EQUAL', 'VAR', 'RIGHT_PARENTHESIS', 'COLON', 'SPACE', 'IF', 'VAR', 'IS', 'VAR', 'COLON', 'SPACE', 'RETURN', 'LEFT_SQUARE_BRACKET', 'VAR', 'FOR', 'VAR', 'IN', 'VAR', 'IF', 'VAR', 'NOT_EQUAL', 'STRING', 'RIGHT_SQUARE_BRACKET', 'ELSE', 'COLON', 'SPACE', 'RETURN', 'VAR', 'ENDL', 'ENDL', 'DEF', 'VAR', 'LEFT_PARENTHESIS', 'VAR', 'COMMA', 'VAR', 'EQUAL', 'VAR', 'RIGHT_PARENTHESIS', 'IS', 'COLON', 'SPACE', 'VAR', 'EQUAL', 'VAR', 'LEFT_PARENTHESIS', 'RIGHT_PARENTHESIS', 'FOR', 'VAR', 'IN', 'VAR', 'COLON', 'SPACE', 'IF', 'VAR', 'IN', 'VAR', 'COLON', 'SPACE', 'YIELD', 'LEFT_SQUARE_BRACKET', 'VAR', 'DOT', 'VAR', 'LEFT_PARENTHESIS', 'RIGHT_PARENTHESIS', 'COMMA', 'VAR', 'RIGHT_SQUARE_BRACKET', 'CONTINUE', 'ENDL', 'FOR', 'VAR', 'COMMA', 'VAR', 'IN', 'VAR', 'COLON', 'SPACE', 'IF', 'VAR', 'DOT', 'VAR', 'LEFT_PARENTHESIS', 'VAR', 'RIGHT_PARENTHESIS', 'COLON', 'SPACE', 'YIELD', 'LEFT_SQUARE_BRACKET', 'VAR', 'COMMA', 'VAR', 'RIGHT_SQUARE_BRACKET', 'BREAK', 'ELSE', 'COLON', 'SPACE', 'RAISE', 'VAR', 'LEFT_PARENTHESIS', 'STRING', 'PERCENT', 'VAR', 'RIGHT_PARENTHESIS', 'YIELD', 'LEFT_PARENTHESIS', 'STRING', 'COMMA', 'STRING', 'RIGHT_PARENTHESIS', 'YIELD']
```

Şekil 2.14 : Moretokenize ile ayrıştırılmış bir Python betiğinin çıktısı

Şekil 2.14 üzerinde görüldüğü üzere “def” ifadesi artık bir isimlendirme değil bir öntanımlı ifade olarak yer alıyor. “def” ifadesinin yanı sıra Python içerisinde bulunan tüm öntanımlı ifadeler (if, raise, except gibi) ayrıştırılabilir ve çıktı olarak yalnızca istenen kolonun verilerini döndürüyor.

2.3.2.2 Levenshtein mesafesi

Levenshtein mesafesi(LM), iki metnin benzerliği karşılaştırmak için kullanılan bir çok algoritmadan yalnızca birisidir. LM, adını kendisini bulan Vladimir Leveshtein’in soyadından almıştır. LM sıfıra ne kadar yakınsa iki metin birbirine o kadar benzerdir. Eğer ki LM sıfır ise iki metnin birbirinin aynısı olduğu anlamına gelmektedir. Doğal dil işleme ve veri madenciliği alanlarında sıklıkla kullanılan bir parametredir. İki metin

arasında LM hesaplamak için iki harfi karşılaştırırken temel olarak üç parametre vardır: eşleşme durumunda verilecek skor, eşleşmeme olduğu durumda verilecek skor, değiştirme olduğunda verilecek skor.

Şekil 2.15 üzerinde eşleşme durumu için “0”, silme ve değiştirme içinse “1” skorunun verildiği “monday” ve “friday” kelimeleri bir LM hesaplaması gerçekleştirilmiştir.

		F	R	I	D	A	Y
	0	1	2	3	4	5	6
M	1	1	2	3	4	5	6
O	2	2	2	3	4	5	6
N	3	3	3	3	4	5	6
D	4	4	4	4	3	4	5
A	5	5	5	5	4	3	4
Y	6	6	6	6	5	4	3

Şekil 2.15 : Monday ve Friday kelimelerinin LD hesaplaması

Şekil 2.15 üzerinde görüldüğü üzere sonuç skoru “3” olarak bulunmuştur. İki kelime uzunlukları küçük uzunluklarda olduklarından dolayı bu fark gözle de görülebilmektedir. “Monday” kelimesinin ilk üç harfi ve “Friday” kelimesinin yine ilk üç harfi birbirlerinden tamamiyle farklıdır. Sonucun üç çıkmasının sebebi ilk üç karakteri değiştirerek veya silerek diğer kelimeyi elde edebiliyor olmamızdır. Sonuç olarak LM bizlere iki metin arasında yapılacak minimum değişiklik ile diğer metni elde edebilmek için gerekli minimum işlem sayısını vermektedir. Bu yüzden ki LM iki farklı kod içeren sınav dosyasının yapısını karşılaştırma konusunda büyük bir kolaylık sağlamaktadır.

2.3.2.3 Graf oluşturma

Betikler ayrıştırılırken elde edilen benzerlik matrisi çalışma zamanında bir depolama yapılmadan direkt olarak bir grafa dönüştürülerek PyNar dosyalarının tutulduğu “chunks/files/exam_medias/<ogretmen_uuuiid>/<sınav_adı>_lev.svg” dizini altına NetworkXX kütüphanesi sayesinde SVG çıktısı şeklinde kaydedilir.

Benzerlik grafindaki “benzerlik” parametresi değişken olduğundan graf oluşturma işlemi öğretmen tarafından istenildiği kadar tekrarlanabilir. Dolaylı olarak betik

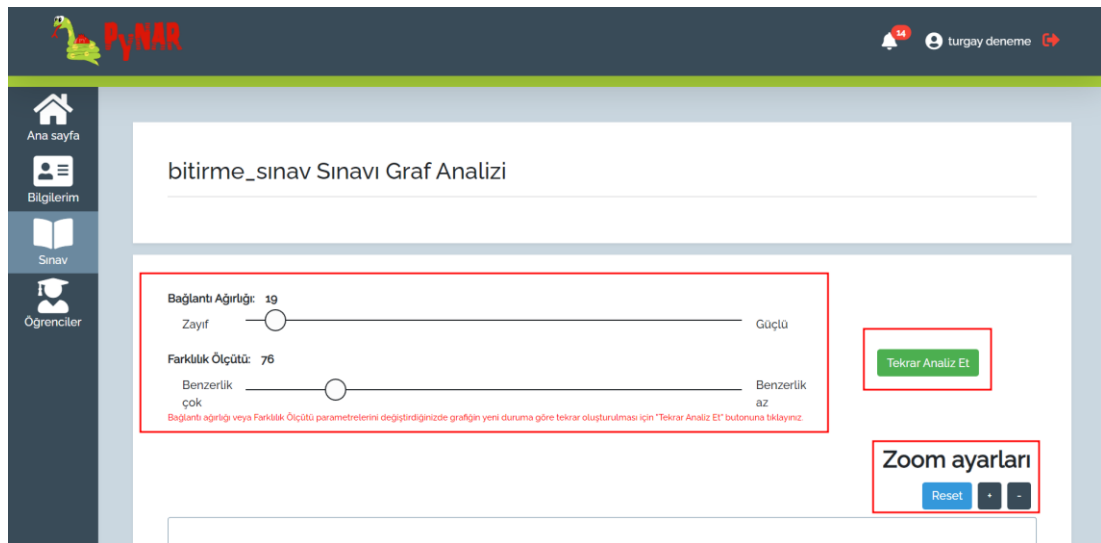
ayırıştırma işlemi de sürekli olarak tekrarlanma eğilimindedir. Arayüz üzerinden benzerlik parametresi seçildikten sonra analiz her tekrarlanışında elde edilen yeni SVG aynı dosya adı ile kaydedilir ve Öğretmen'e sunulur.

2.3.2.4 Uygulama

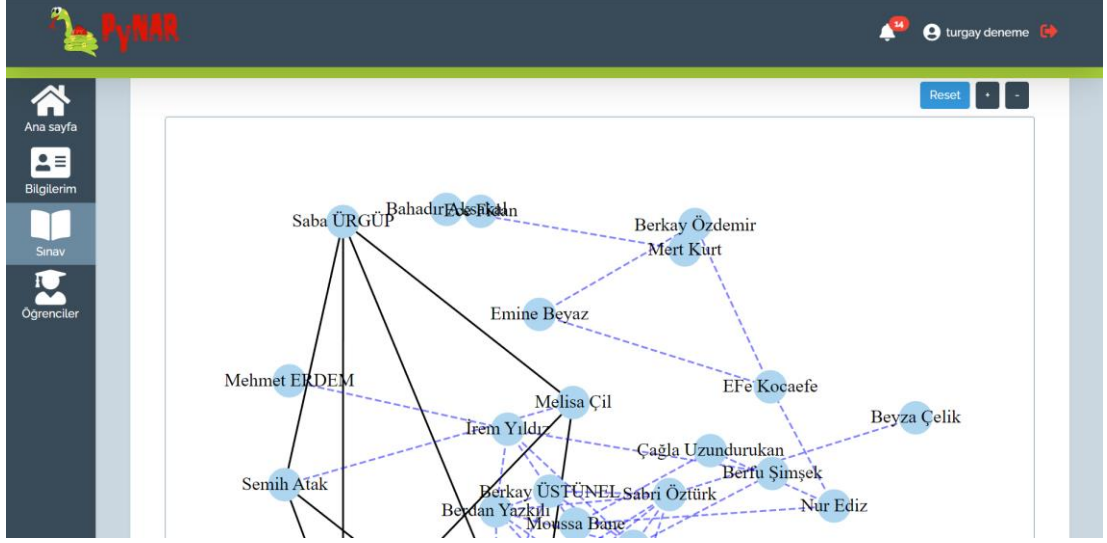
PyNar sistemi üzerinde benzerlik grafi analizini Pynar Web Portal'ı üzerinden gerçekleştirebilmek için sırasıyla aşağıdaki adımlar uygulanmalıdır:

- Adım 1: Öğretmen hesabına giriş yapılır
- Adım 2: Sol menüden “Sınav” sekmesine tıklanır
- Adım 3: Sınav sekmesinde analiz yapılmak istenen sınav satırı üzerinde “sonuçlar” butonuna tıklanır
- Adım 4: Açılan pencerede “Sınavı Analiz Et” adlı butona tıklanır
- Adım 5: Açılan pencerede ortalama süre analizi penceresi içerisinde “Sınavı analiz et” butonuna tıklanır
- Adım 6: Açılan pencere içerisinde benzerlik grafi görülecektir

Yukarıdaki adımlar sırasıyla tamamlandıktan sonra açılmış olan en son pencere içerisinde benzerlik grafi görülecektir. Benzerlik grafinin üst başlık kısmında ise yakınlaştırma ayarları ve en üstte benzerlik parametre ayarları bulunmaktadır. Adım 1'den Adım 5'e kadar ve Adım 5 dahil bir şekilde Şekil 2.7'den başlayarak sırasıyla 5 şekil ile verilmiştir. Aşağıda Şekil 2.16 ve Şekil 2.17 ile Adım 6 verilmiştir.



Şekil 2.16 : Benzerlik parametreleri, tekrar analiz butonu ve yakınlaştırma ayarları



Şekil 2.17 : Benzerlik grafi örneği (isimler gerçek kişilerden bağımsızdır)

2.4 PyNar Hata Analizi

Bir fikri koda dökerken sürekli olarak programlama dilinin ifadelerine uygun kelimeler yazarız. Çoğu zaman kullandığımız editör hatalar, uyarılar vb. mesajlar yardımıyla bizlere sinyaller gönderir. Biz de bu sinyallere bakarak sorunlarımızı gidermeye çalışırız. Ancak bir hata türü vardır ki editörler günümüzde bunları hala düzeltememekte ve yalnızca çalışma zamanlarında sinyaller göndermektedir. Bu sinyaller sayesinde yine çeşitli aksiyonlar alınarak hatalar düzeltilmektedir.

Günümüz yazılım mühendislerinin en büyük problemlerinden birisi yüksek kullanılabilirliğe sahip sistemler planlamak ve bunları satırlara dökerek gerçek bir şekilde uygulamaktır. Kullanılabilirliğe yüksek bir sistem tasarlayabilmenin yolu güvenilir kod yazmaktan geçmektedir.

Güvenilir bir kod yazmak yüklü bir hata tecrübesine sahip olmayı gerektirir ve sistem ne kadar büyürse tecrübe o kadar kritik önem kazanır. Hatasız kod yazmak imkansız yakın bir kusursuz deha ve kusursuz planlama gerektirir. Mantıksal hatalar her zaman olacaktır ancak çalışma zamanı hataları tecrübe ile engellenebilmektedir.

PyNar Kod Editörü, öğrencilerin yaptıkları çalışma zamanı hatalarını yakalar ve onlara kendi tecrübelerinden faydalanmalarını sağlayarak çeşitli çözüm önerilerinde bulunur. PyNar Chatbot büyük bir veri seti ile eğitildiği için yüklü bir tecrübeye dolaylı olarak da güvenilir bir doğruluk oranına sahiptir. Ancak PyNar Chatbot anlık olarak öğrenciye bilgilendirme yapar dolayısıyla ondan bir öğretmen gibi davranıp konu tekrarı yapması beklenemez.

PyNar Kod Editörü, öğrencilere hatalarını düzeltebilmek için öneriler yapmakla kalmayarak, öğrencilerin sınav esnasında gerçekleştirdikleri hataları günlükler üzerine kayıtlar. Öğrencilerin yaptıkları her hata sınav sonuçlarıyla birlikte günlükler içerisinde uzak sunuculara yüklenmektedirler.

Öğretmenlerin, öğrencilerinin sınav esnasında yaptıkları hataları inceleyebilmeleri için “PyNar Hata Analizi” geliştirilmiştir. Geliştirilen analiz ile öğrencilerin ortak olarak en çok yaptıkları hatalar PyNar Web Portal üzerinden erişilebilir hale getirilmektedir. Bu bağlamda, öğretmenler yapılan hataları analiz ederler ve bu hatalara ilişkin olarak öğrencilerin gelişimlerini eniyileleyebilmektedirler.

2.4.1 Uygulama

PyNar sistemi üzerinde ortak hata analizini Pynar Web Portal’ı üzerinden gerçekleştirebilmek için sırasıyla aşağıdaki adımlar uygulanmalıdır:

- Adım 1: Öğretmen hesabına giriş yapılır
- Adım 2: Sol menüden “Sınav” sekmesine tıklanır
- Adım 3: Sınav sekmesinde analiz yapılmak istenen sınav satırı üzerinde “sonuçlar” butonuna tıklanır
- Adım 4: Açılan pencerede “Sınavı Analiz Et” adlı butona tıklanır
- Adım 5: Açılan pencerede ortalama süre analizi penceresi içerisinde “En çok yapılan hataları analiz et” butonuna tıklanır
- Adım 6: Açılan pencere içerisinde bir tablo üzerinde tüm hata listesi görüntülenecektir

Hata analizinin üst başlık kısmında ise yakınlaştırma ayarları ve en üstte benzerlik parametre ayarları bulunmaktadır. Adım 1’den Adım 5’e kadar ve Adım 5 dahil bir şekilde Şekil 2.7’den başlayarak sırasıyla 5 şekil ile verilmiştir. Aşağıda Şekil 2.18 ile Adım 6 verilmiştir.



The screenshot shows the PyNAR system interface. The top header includes the PyNAR logo and a user profile section with a notification bell and the text 'turgay deneme'. The left sidebar contains navigation icons for 'Ana sayfa', 'Bilgilerim', 'Sınav', and 'Öğrenciler'. The main content area is titled 'bitirme_sınav Sınavı Hata Analizi'. Below the title, there is a section 'Öğrencilerin en çok yaptığı hatalar' with a dropdown menu set to '10' and a search bar. The table below lists the errors and their frequencies.

Hata mesajı	Tekrarlanma sayısı
(ifadesi kapatılmadı	70
Ifadeler satırsonu veya noktalı virgülle ayrılmalıdır	39
d ifadesi tanımlı değil	24
: ifadesi bekleniyor	20
tizme ifadesi tanımlı değil	20
timXe içe aktarmı çözemedi	20
aslan ifadesi tanımlı değil	15

Şekil 2.18 : Adım 6: PyNar ortalama hata analizi

3. SONUÇ

Geliştirilen PyNar Yargıç analiz aracı sayesinde sınav esnasında aldatmaya yönelik hareketlerin tespit edilebilmesinin önü açılmış bulunmaktadır. Elde edilmiş günlük ve sınav cevap dosyalarını işleyerek öğrencilerin birbirleri aralarında veya kendi başlarına hile gerçekleştirmelerinin tespit edilmesi konusunda PyNar Kod Editörünün sınav özelliği güçlü ve tekil bir yön sağlanmıştır.

PyNar Hata Analizi aracı sayesinde öğrencilerin kişisel gelişimine katkıda bulunmak adına PyNar Kod Editörüne farklı bir bakış açısı kazandırılmıştır. Öğrenciler kodlarındaki çalışma zamanı hatalarını tecrübesinden yararlandıkları PyNar Chatbot ile düzeltmekle kalmayıp öğretmenlerinin yönlendirmeleriyle neden bu hataları yaptıklarını derin bir şekilde inceleyebilme ve hatalarını tekrar eğitim yoluyla düzeltebilme imkanı elde etmiş olacaktırlar.

3.1 Öneriler

Büyük Veri her zaman bir çözüm ortaya koymaya çalışan sistemlerin sorunu olmuştur. PyNar Yargıç ve PyNar Hata Analizi de Büyük Veri sorunuyla karşı karşıya kalmaktadır. Analiz sistemleri ortalama bir sınıf (80 kişilik bir sınıf) üzerinde sorunsuz bir çalışma ortaya koyabilecektir. Ancak sınıf mevcudu büyüdükçe sınav analizi gerçekleştirmek çok daha zorlu bir hale gelecektir. Bu sorunun çözümü için yapılabilecek major değişiklikler:

- Sistemdeki servisler birbirlerinden ayrılarak tam anlamıyla Servis Odaklı Mimariye geçiş yapılabilir
- Sistem Mimarisi üzerinde görüldüğü üzere tek bir MySQL deposu bulunmaktadır. Birden fazla ve master-slave ilişkisine sahip olacak şekilde kurgulanabilir.
- Sistem önüne bir yük dengeleyici eklenerek yük dağıtılabilir ve esneklik kazandırılabilir

Minor değişiklikler:

- Çoktan seçmeli, doğru-yanlış gibi farklı sınav tipleri eklenebilir

- Öğretmenlere analiz konusunda yardımcı olarak otomatize edilmiş bir takım Derin Öğrenme algoritmaları eklenebilir
- LM yerine farklı bir eşleştirme yöntemi kullanılabilir veya LM optimize edilebilir

Gürültü oluşturabilecek öğrenciler için bir tespit algoritması kullanılabilir. Örneğin sürekli bir şekilde PyNar Kod Editörüne karakterler yazıp bunları silen bir öğrenci muazzam bir günlüğe sahip olacaktır ancak ortada kayda değer bir kod olmayacaktır. Bu şekilde gürültü yaratan öğrencileri tespit ederek öğretmene yeni bir analiz mekanizması sunulabilir, direkt olarak bu öğrenciler analiz dışı bırakılabilir veya bu öğrenciler kırmızı renkle işaretlenebilir.

KAYNAKLAR

- Balcioğlu, B.**, (2015). *An improved spatio-relational database design for urban conservation and its performance analysis* (Yüksek Lisans Tezi). Orta Doğu Teknik Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.
- Bolat, A. N.**, (2019). *NoSQL ve RDBSM yapıların performans karşılaştırması* (Yüksek Lisans Tezi). Beykent Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.
- Byun, J., Park, J., & Oh, A.**, (2020). Detecting contract cheaters in online programming classes with keystroke dynamics. In Proceedings of the Seventh ACM Conference on Learning@ Scale, 273-276.
- Ceresnak, R., Chovancova, O.**, (2019). Comparison of Query Performance Between MySQL and MongoDB Database. *Central European Researchers Journal*, 5, 45-51.
- Györödi C. A., Dumşeu-Burescu D. V., Zmaranda D. R., Györödi R.Ş.**, (2022). A Comparative Study of MongoDB and Document-Based MySQL for Big Data Application Data Management. *Big Data and Cognitive Computing*. 6(2):49. <https://doi.org/10.3390/bdcc6020049>
- Hammood, H. A.**, (2016). *A comparison of NoSQL database systems: a study on MongoDB, Apache HBASE and Apache Cassandra* (Yüksek Lisans Tezi). Çankaya Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.
- Kızıllırmak, E.**, (2020). *İngilizce-Türkçe çeviri metinlerde Levenshtein uzaklığı ile desteklenmiş çapa tabanlı cümle eşleme* (Yüksek Lisans Tezi). Maltepe Üniversitesi, Lisansüstü Eğitim Enstitüsü, İstanbul.
- Koçer, S.**, (2019). *İlişkisel veri tabanlarının uygulandığı araç takip sisteminin büyük veriye uyarlanması* (Yüksek Lisans Tezi). Maltepe Üniversitesi, Fen Bilimleri Enstitüsü, İstanbul.
- Kustanto, C., & Liem, I.**, (2009). Automatic source code plagiarism detection. In 2009 10th ACIS International conference on software engineering, artificial intelligences, networking and parallel/distributed computing, 481-486, IEEE.
- MongoDB vs MySQL Differences**. Erişim: 23 Haziran 2022, <https://www.mongodb.com/compare/mongodb-mysql>
- Obadi, G., Drázdilová, P., Martinovic, J., Slaninová, K., & Snásel, V.**, (2010). Using Spectral Clustering for Finding Students' Patterns of Behavior in Social Networks. In DATESO, 118-130.
- Pawelczak, D.**, (2018). Benefits and drawbacks of source code plagiarism detection in engineering education. IEEE Global Engineering Education Conference (EDUCON), 1048-1056. IEEE.
- Taha, A. I.**, (2017). *A comprehensive comparison of NoSQL and relational database management systems* (Yüksek Lisans Tezi). Çankaya Üniversitesi, Fen Bilimleri Enstitüsü, Ankara.
- Tiong, L. C. O., & Lee, H. J.**, (2021). E-cheating Prevention Measures: Detection of Cheating at Online Examinations Using Deep Learning Approach - A Case Study. arXiv, preprint arXiv:2101.09841, <https://doi.org/10.48550/arXiv.2101.09841>

Tufan, Ö., (2008). *Improving the quality of the Turkish address records by using Levenshtein distance algorithm* (Yüksek Lisans Tezi). Bahçeşehir Üniversitesi, Lisansüstü Eğitim Enstitüsü, İstanbul.

Uzunbayır, S., (2015). *A comparison between relational database models and NoSQL trends on big data design challenges using a social shopping application* (Yüksek Lisans Tezi). İzmir Ekonomi Üniversitesi, Fen Bilimleri Enstitüsü, İzmir.

ÖZGEÇMİŞ

TARANMIŞ
VESİKALIK
FOTOĞRAF

Ad-Soyad : Fatih ATEŞ
Doğum Tarihi ve Yeri : 27/09/2000, Bursa
E-posta : fatiates@gmail.com

BİTİRME ÇALIŞMASINDAN TÜRETİLEN MAKALE, BİLDİRİ VEYA SUNUMLAR:

-
-
-