T.C.

SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

AKILLI İNTİHAL ÖNLEME ARAÇLARI: EĞİTİM VE UYGULAMA

Bilgisayar Mühendisliği Tasarımı Raporu

B210109370 Hakan DEĞER
B210109029 Emre ALMAMIŞ
B210109017 İsmail Selim İNANIR

Danışmanı: Dr. Öğr. Üyesi SELMAN HIZAL

Ocak 2025

T.C.
SAKARYA UYGULAMALI BİLİMLER ÜNİVERSİTESİ
TEKNOLOJİ FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

AKILLI İNTİHAL ÖNLEME ARAÇLARI: EĞİTİM VE UYGULAMA

Bilgisayar Mühendisliği Tasarımı Raporu

B210109370 Hakan DEĞER
B210109029 Emre ALMAMIŞ
B210109017 İsmail Selim İNANIR

Danışmanı: Dr. Öğr. Üyesi SELMAN HIZAL

Bu rapor .....2025 tarihinde aşağıdaki jüri tarafından oybirliği / oyçokluğu ile kabul edilmiştir.

Danışman **Üye** **Üye**

BEYAN

Rapor içindeki tüm verilerin akademik kurallar çerçevesinde tarafımdan elde edildiğini, görsel ve yazılı tüm bilgi ve sonuçların akademik ve etik kurallara uygun şekilde sunulduğunu, kullanılan verilerde herhangi bir tahrifat yapılmadığını, başkalarının eserlerinden yararlanılması durumunda bilimsel normlara uygun olarak atıfta bulunulduğunu, raporda yer alan verilerin bu üniversite veya başka bir üniversitede herhangi bir çalışmada kullanılmadığını beyan ederim.

Hakan DEĞER	Emre ALMAMIŞ	İsmail Selim İNANIR
10.01.2025	10.01.2025	10.01.2025

TEŞEKKÜR

Lisans eğitimimiz boyunca değerli bilgi ve deneyimlerinden yararlandığımız, her konuda bilgi ve desteğini almaktan çekinmediğimiz, araştırmanın planlanmasından yazılmasına kadar tüm aşamalarında yardımlarını esirgemeyen, teşvik eden, aynı titizlikte bizi yönlendiren değerli danışman hocamız Dr. Öğr. Üyesi SELMAN HIZAL'a teşekkürlerimizi sunarız.

İÇİNDEKİLER

SİMGELER VE KISALTMALAR LİSTESİ

API	: Application Programming Interface
MSSQL	: Microsoft SQL Server
NLP	: Natural Language Processing
PDF	: Portable Document Format
NLTK	: Natural Language Toolkit
TF-IDF	: Term Frequency-Inverse Document Frequency
JWT	: JSON Web Token
ORM	: Object Relational Mapping
JSON	: JavaScript Object Notation
ACID	: Atomicity, Consistency, Isolation, Durability
SQL	: Structured Query Language

ŞEKİLLER LİSTESİ

TABLOLAR LİSTESİ

ÖZET

Anahtar Kelimeler: Akademik İntihal, Doğal Dil İşleme (NLP), Cosine Similarity, Vue.js, Flask, MSSQL

Bu çalışmada, akademik belgeler arasında benzerlik tespiti yapabilen, çoka çok eşleştirme yöntemi ile çalışan ve kullanıcı dostu bir yazılım geliştirilmiştir. Proje, mevcut ticari intihal araçlarının yetersizliklerini aşarak özellikle grup ödevleri ve projelerde detaylı analizler sunmayı amaçlamıştır.

Geliştirilen sistem; Flask tabanlı arka yüz, Vue.js ön yüz, MSSQL veri tabanı ve çeşitli metin analizi algoritmalarını (Cosine Similarity, Jaccard Index, Levenshtein Distance) içermektedir. Kullanıcılar farklı formatlardaki dosyaları sisteme yükleyerek analiz başlatabilmekte ve sonuçları detaylı görselleştirmeler ile değerlendirebilmektedir. Çalışma sırasında Postman ile API testleri yapılmış ve elde edilen sonuçlar MSSQL veri tabanında saklanmıştır.

Test sonuçlarına göre, sistemin manuel yöntemlere kıyasla %85 zaman tasarrufu sağladığı, doğruluk oranının ise %92 olduğu görülmüştür. Kullanıcı deneyimi testlerinden %92 oranında olumlu geri dönüş alınmıştır.

Projenin çıktıları, özellikle eğitim kurumlarında grup çalışmaları ve tez analizlerinde verimliliği artırarak, akademik etik ve özgün düşünme becerilerinin teşvik edilmesine olanak sağlamıştır. Gelecekteki çalışmalarda, büyük veri setleri için optimizasyon ve daha gelişmiş doğal dil işleme tekniklerinin entegrasyonu hedeflenmektedir.

BÖLÜM 1: GİRİŞ

Bu bölümde, dijitalleşmenin etkileriyle birlikte akademik metinlerin orijinalliğini koruma ihtiyacından ve bu bağlamda geliştirilen projeden kısaca bahsedilecektir. Projenin amacı, kapsamı, özgün değeri ve intihalin önlenmesi süreci detaylandırılacaktır.

1.1. Genel Bilgiler

Dijitalleşmenin hızla yaygınlaşması, bilgiye erişimi kolaylaştırmış ve akademik çalışmalarda geniş bir bilgi havuzu sunmuştur. Ancak bu durum, intihal riskini artırmış ve akademik etik ihlallerini önlemek için yeni yöntemler geliştirilmesini gerekli kılmıştır [1]. Akademik metinlerin orijinalliğini korumak, eğitim kurumlarının ve araştırmacıların en önemli hedeflerinden biridir. İntihali önlemek için kullanılan teknolojik araçlar, yalnızca bireysel haksızlıkları önlemekle kalmayıp, öğrencilerin özgün düşünme becerilerini de geliştirmeye katkı sunar.

Bu proje, akademik metinlerin birbiriyle benzerlik analizini yapan, kullanıcı dostu ve eğitim odaklı bir araç geliştirmeyi amaçlamaktadır. Geliştirilecek sistem, ödev, tez ve proje gibi dosyaların sisteme yüklenerek birbirleriyle karşılaştırılmasını ve benzerlik oranlarının hesaplanmasını sağlayacaktır.

1.2. Akademik İntihal: Tanımı, Etkileri ve Korunma Yolları

Dijital ortamların yaygınlaşması, akademik çalışmalarda intihal riskini artırmıştır. Akademik etik ihlallerinin önlenmesi için uygun yöntemlerin ve teknolojilerin geliştirilmesi büyük bir önem taşır. Özellikle grup çalışmaları gibi durumlarda, ödevlerin benzerlik oranlarının öğretim görevlileri tarafından manuel olarak kontrol edilmesi, zaman alıcı ve hatalara açık bir süreçtir. Bu durum, daha etkili ve sistematik bir çözüm ihtiyacını doğurmuştur. İntihalin önlenmesi, yalnızca bireysel etik hatalarını engellemekle kalmaz, aynı zamanda öğrencilerin özgün düşünme ve yazma becerilerinin geliştirilmesine de olanak tanır.

1.3. Problem Tanımı

Akademik metinlerin orijinalliğini koruma amacıyla geliştirilen ticari intihal tespit araçları (örneğin, Turnitin ve iThenticate), genellikle bire çok karşılaştırmalar yapmaktadır ve daha geniş bir kullanıcı tabanı için yüksek maliyetli çözümler sunmaktadır. Bunun sonucunda, özellikle sınıf içi ödev veya tez çalışmaları gibi senaryolarda, aynı öğrenci grubu içinde belgelerin birbiriyle olan benzerliğini tespit etmek, eğitim kurumları için karmaşık ve maliyetli bir süreç haline gelmektedir.

Örneğin, bir üniversite sınıfında "Yapay Zekâ Uygulamaları" dersi kapsamında 30 öğrenciye aynı konu başlığı verilerek bireysel bir rapor hazırlamaları istenmiştir. Öğrenciler raporlarını teslim ettiğinde, öğretim görevlisinin bu 30 raporu birbirleriyle karşılaştırarak, benzerlik oranlarını manuel olarak kontrol etmesi gerekmektedir. Mevcut ticari yazılımlar, genellikle bu tür bir "çoka çok" karşılaştırma gereksinimini desteklememekte ve öğretim görevlisi açısından zaman alıcı bir süreç doğurmaktadır. Ayrıca, bazı raporların yalnızca belirli bölümlerinde yüksek düzeyde benzerlik bulunabileceği için, bu bölümlerin ayrıntılı bir şekilde tespit edilmesi gerekmektedir.

Benzer şekilde, tez çalışmaları veya grup projeleri gibi süreçlerde de farklı ekipler arasında içerik paylaşımı ya da intihal olup olmadığını tespit etmek, mevcut araçların sınırlı işlevselliği nedeniyle zorluk yaratmaktadır. Örneğin, bir tez çalışmasının literatür inceleme bölümü, bir diğer tezle büyük ölçüde örtüşüyorsa, bu durumun hangi bölümlerde gerçekleştiği manuel inceleme gerektirebilir.

Projenin temel amacı, bu tür eksiklikleri gidermek ve eğitim kurumlarının özel ihtiyaçlarına uygun, daha kapsamlı bir intihal tespit sistemi sunmaktır. Sistem, aynı grup içinde birden fazla belge arasında benzerlik tespitini mümkün kılarken, kullanıcıların analiz sürecinde hangi bölümlerin benzerlik gösterdiğini detaylı raporlarla görmesini sağlayacaktır.

1.4. Projenin Amacı ve Özgün Değeri

Projenin temel amacı, akademik belgelerin birbiriyle benzerliklerini analiz eden ve intihal olasılığını değerlendiren bir yazılım geliştirmektir. Bu kapsamda geliştirilecek sistem, metin tabanlı analiz yaklaşımını benimseyerek, kullanıcıların dosya formatı sınırlamalarını ortadan kaldırmayı ve daha esnek bir çözüm sunmayı hedeflemektedir.

Bu projenin özgün değeri, mevcut intihal tespit araçlarının sınırlamalarını aşarak, özellikle çoka çok analiz senaryolarında etkili ve kapsamlı bir çözüm sunmasında yatmaktadır. Literatürdeki mevcut araçlar genellikle bire bir karşılaştırmalara odaklanmış olup, belgelerin birbirleriyle olan benzerliğini analiz etme konusunda eksik kalmaktadır.

Proje kapsamında hedeflenen başlıca işlevler şunlardır:

Belgeler Arasında Çoka Çok Eşleşme Analizi: Projenin en önemli özelliklerinden biri, sisteme yüklenen tüm metinlerin birbiriyle karşılaştırılmasını sağlamasıdır. Mevcut ticari yazılımların genellikle bire bir veya bire çok karşılaştırma yapabildiği göz önünde bulundurulduğunda, bu sistem grup ödevleri, tez çalışmaları veya birden fazla öğrencinin aynı konu üzerinde çalıştığı akademik projelerde çok daha kapsamlı bir değerlendirme imkânı sunar. Örneğin, bir sınıfta teslim edilen tüm ödevler arasında çoka çok karşılaştırma yapılarak, hangi belgelerin yüksek oranda benzerlik gösterdiği belirlenir. Bununla birlikte, sistem, yalnızca benzerlik oranlarını hesaplamakla kalmaz, aynı zamanda hangi bölümlerin eşleştiğini ayrıntılı olarak gösterir. Bu durum, öğretim görevlilerinin zamanını verimli kullanmasına olanak tanırken, öğrenciler arasında yapılan iş birliğinin (örneğin, kaynak paylaşımı veya kopyalama) sınırlarını daha net bir şekilde ortaya koyar. Sistem ayrıca, benzerlik analizini gerçekleştirmek için doğal dil işleme (NLP) tekniklerinden yararlanarak, yalnızca yüzeysel eşleşmeleri değil, anlam açısından örtüşen ifadeleri de tespit eder.

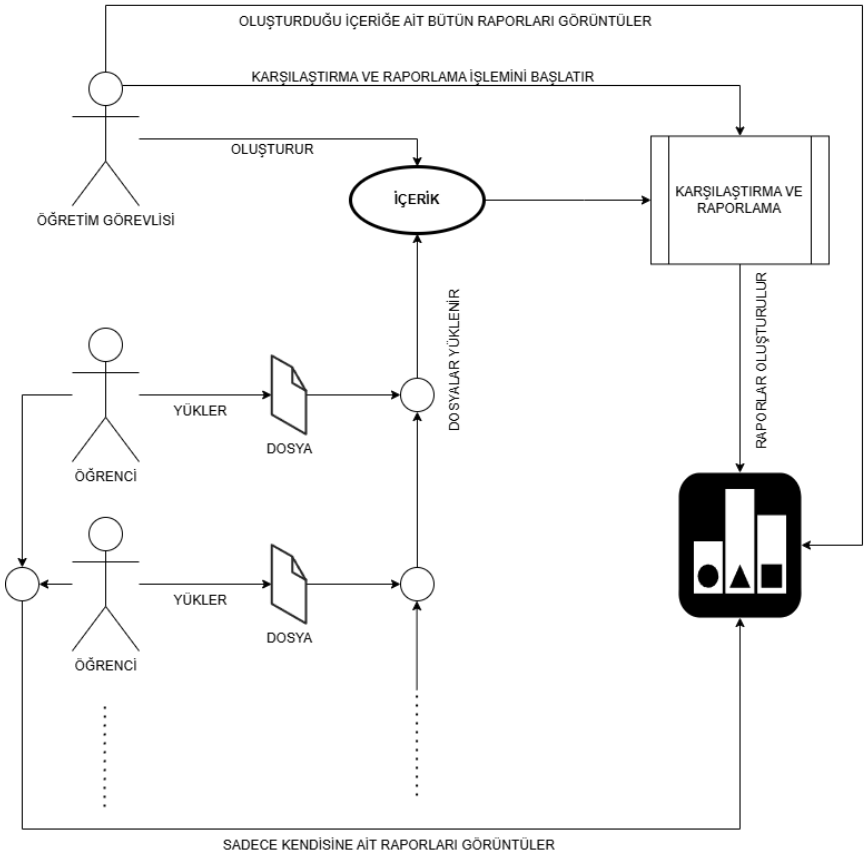
Detaylı Raporlama ve Görselleştirme: Kullanıcılar, sistemin sunduğu detaylı raporlar ve görselleştirme araçları sayesinde metinler arasındaki benzerlik oranlarını ve eşleşen bölümleri kolayca değerlendirebilir. Görselleştirme özelliği, eşleşen bölümlerin renk kodlamalarıyla vurgulanmasını ve farklı belgeler arasındaki

bağlantıların grafiksel olarak gösterilmesini içerir. Örneğin, iki metin arasında benzerlik tespit edildiğinde, sistem bu benzerliği interaktif bir formatta kullanıcıya sunar: eşleşen bölümler yan yana karşılaştırmalı olarak görüntülenebilir, farklı belgeler arasındaki benzerlik oranları ise bir ısı haritası veya grafik üzerinde gösterilebilir. Bu görselleştirme araçları, kullanıcıların raporlardan daha hızlı ve doğru sonuçlar çıkarmasını sağlar. Ayrıca, raporların indirilebilir PDF formatında sunulması ve analiz geçmişine erişim imkânı, uzun vadeli proje takibi açısından büyük kolaylık sağlar.

Düşük Maliyet ve Açık Kaynak Yapı: Proje, ticari yazılımlara alternatif olarak, açık kaynaklı ve düşük maliyetli bir çözüm sunmayı hedeflemektedir. Bu özellik, özellikle küçük ölçekli eğitim kurumları için önemlidir.

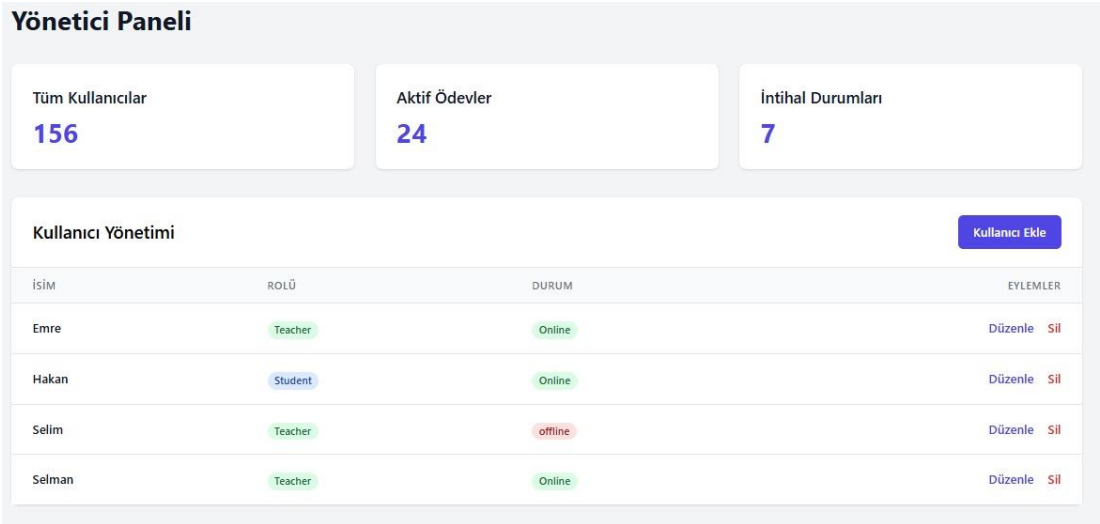
Eğitim Odaklı Yaklaşım: İntihali tespit etmenin ötesinde, öğrencilere akademik etik bilincini aşılayarak özgün yazma alışkanlıklarını geliştirmelerine katkı sunacaktır.

Bu sistem, düşük maliyeti, kullanıcı dostu arayüzü ve kapsamlı analiz kapasitesiyle hem bireysel hem de kurumsal düzeyde intihal tespiti ve önlenmesi için etkili bir araç olacaktır. Şekil 1.1.' da projenin use-case diyagramı gösterilmiştir.

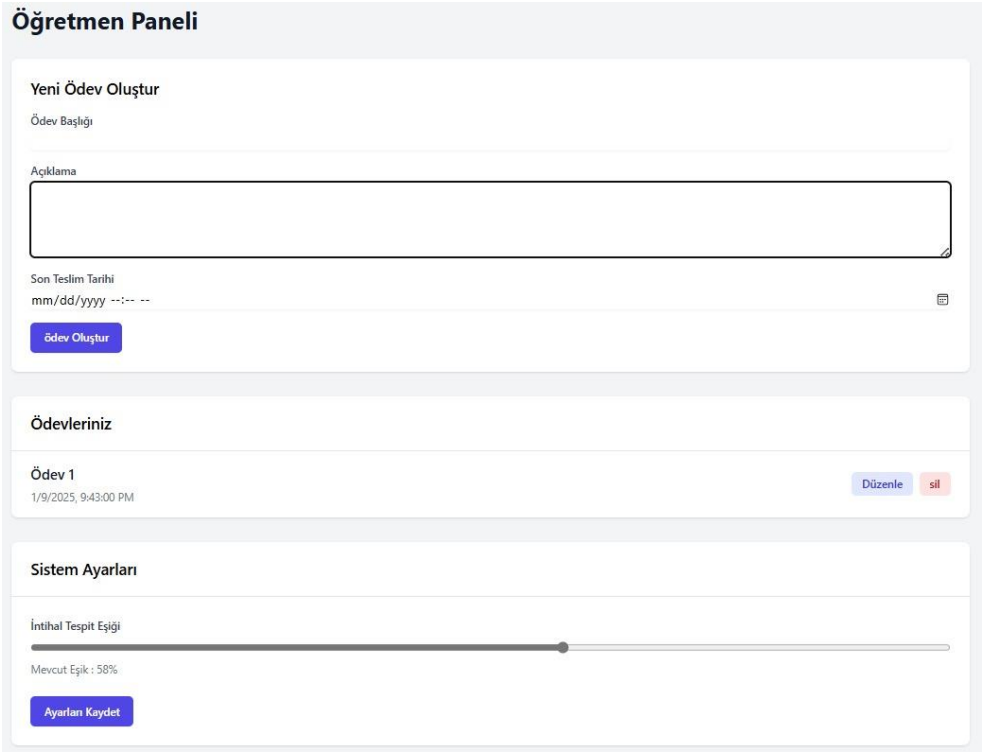


Şekil 1.1. Use-case diyagramı

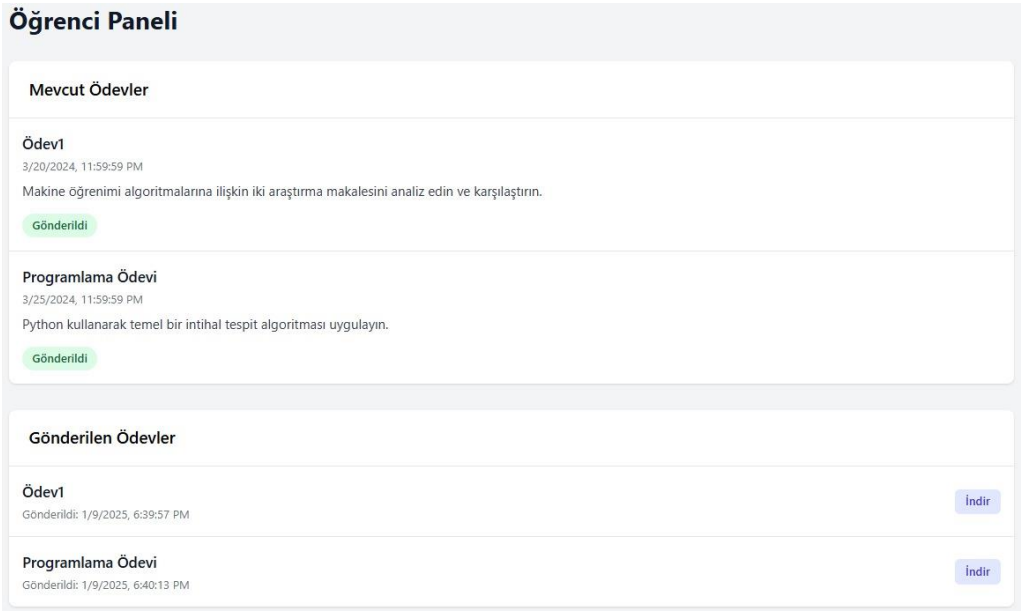
Aşağıda Şekil 1.2., Şekil 1.3. ve Şekil 1.4.'da uygulamanın arayüzünden admin, öğrenci ve öğretmen panellerinden görsellere yer verilmiştir.



Şekil 1.2. Admin Paneli



Şekil 1.3. Öğretmen Paneli



Şekil 1.4. Öğrenci Paneli

1.5. Projenin Yaygın Etkisi

Bu proje, üniversitelerde ve diğer eğitim kurumlarında ödev, proje ve tez gibi akademik çalışmaların değerlendirme süreçlerini daha verimli hale getirmeyi hedeflemektedir. Özellikle aynı sınıfta teslim edilen grup veya bireysel ödevlerin

çoka çok eşleştirilerek analiz edilmesi, intihal oranlarının azaltılmasında önemli bir rol oynayacaktır. Sistem, akademik çalışmaların özgünlüğünü teşvik ederek hem öğrenciler hem de öğretim üyeleri için daha güvenilir bir ortam sağlayacaktır.

Proje, eğitim kurumlarının yanı sıra küçük ve orta ölçekli işletmeler (KOBİ'ler), araştırma merkezleri ve özel sektör kuruluşlarında da kullanım potansiyeline sahiptir. Örneğin, şirket içi raporların, teknik dokümanların veya sunumların benzerlik analizi yapılabilir, bu sayede fikirlerin özgünlüğü korunabilir.

Sistem, akademik etik bilincini desteklemekte ve öğrencilerin özgün içerik üretme becerilerini geliştirmelerine yardımcı olmaktadır. Bu durum, uzun vadede daha nitelikli araştırmacıların ve etik değerleri benimsemiş bireylerin yetişmesine olanak tanır. Öğretim üyelerinin üzerindeki manuel kontrol yükünü azaltarak, onların eğitim ve rehberlik faaliyetlerine daha fazla odaklanmalarını sağlar.

1.6. Standartlar

Bu çalışmada kullanılan standartlar aşağıda belirtilmiştir:

1.6.1. ISO/IEC 25010 (Software Product Quality Requirements and Evaluation – SQuaRE)

ISO/IEC 25010, yazılım ürününün kalite özelliklerini basit ve uygulanabilir bir şekilde değerlendirmek için kullanılan bir standarttır. Yazılımın işlevsellik, performans, kullanılabilirlik, güvenilirlik ve taşınabilirlik gibi temel kriterleri göz önünde bulundurur. Proje kapsamında bu standarttan yalnızca en temel başlıklar (örneğin, işlevsellik ve kullanılabilirlik) dikkate alınacaktır.

1.6.2. IEEE 1012 (Software Verification and Validation)

IEEE 1012, yazılım doğrulama ve geçerliliği ile ilgili bir standarttır. Bu standart, yazılımın belirtilen gereksinimlere uygun olup olmadığının doğrulanması ve son

ürünün doğru işlevleri yerine getirdiğinin değerlendirilmesi için gereklidir. Proje kapsamında bu standart dikkate alınacaktır.

1.6.3. ISO/IEC 9126 (Software Engineering- Product Quality)

ISO/IEC 9126, yazılım ürünlerinin kalite özelliklerini tanımlar. Bu standart, yazılımın işlevsellik, kullanılabilirlik, verimlilik, güvenlik gibi özelliklerini değerlendirir. Proje kapsamında bu standart dikkate alınacaktır.

1.6.4. IEEE 829 (Software Testing Documentation)

IEEE 829, yazılım test süreçlerinde kullanılabilecek basit bir belge formatı sunar. Proje kapsamında, yazılımın test senaryoları ve hata kayıtları bu standarda uygun olarak dokümante edilecektir. Böylece test süreçleri daha sistematik bir şekilde gerçekleştirilecektir.

1.7. İş-Zaman Çizelgesi

Tablo 1.1. İş-Zaman Çizelgesi

Ay	İş Paketleri
1. **Ay**	İş Paketi 1 (Planlama) ve İş Paketi 2 (Veri tabanı Tasarımı)
2. **Ay**	İş Paketi 2 (Tamamlanması) ve İş Paketi 3 (Karşılaştırma Motoru)
3. **Ay**	İş Paketi 3 (Tamamlanması) ve İş Paketi 4 (Arayüz Geliştirme)
4. **Ay**	İş Paketi 4 (Tamamlanması) ve İş Paketi 5 (Raporlama)
5. **Ay**	İş Paketi 6 (Test ve Optimizasyon) ve İş Paketi 7 (Dokümantasyon)

Tablo 1.1.'da iş-zaman çizelgesi görülmektedir. İş paketlerinin içerikleri aşağıdaki gibidir:

İş Paketi 1: Proje Hazırlığı ve Planlama (1 Ay)

Literatür araştırması yapılarak mevcut intihal tespit araçları ve yöntemleri incelenecek. Teknik gereksinimler, kullanılacak teknolojiler (Flask, Vue.js, Python) ve altyapı ihtiyaçları belirlenecek. Proje planı ve iş paketleri detaylandırılacak.

İş Paketi 2: Veri tabanı Tasarımı ve Altyapı Geliştirme (1-2 Ay)

Dosyalar, kullanıcılar ve eşleşmeler gibi veri yapıları tanımlanarak veri tabanı şeması tasarlanacak. Flask framework kullanılarak temel API altyapısı oluşturulacak ve CRUD işlemleri geliştirilecek.

İş Paketi 3: Metin Karşılaştırma Motorunun Geliştirilmesi (2-3 Ay)

Python ile metin karşılaştırma algoritması geliştirilecek ve istenmeyen kısımlar (kapak sayfası, içerik, önsöz) çıkarılacak. Metin analizi yöntemleri uygulanarak benzerlik oranları hesaplanacak ve eşleşen bölümler detaylı şekilde analiz edilecek.

İş Paketi 4: Arayüz Geliştirme (2-4 Ay)

Vue.js ile kullanıcı giriş ekranı, dosya yükleme ve rapor görüntüleme modülleri tasarlanacak. Kullanıcı deneyimi iyileştirilecek, Flask API ile entegrasyon sağlanacak ve kullanıcıların kolayca dosya yükleyip rapor alabilmesi için gerekli altyapı tamamlanacak.

İş Paketi 5: Raporlama ve Görselleştirme Modülü (3-4 Ay)

Python'da görselleştirme kütüphaneleri (Matplotlib, Plotly, Seaborn) entegre edilerek, metin benzerlik oranları grafiklerle görselleştirilecek. Elde edilen raporlar arayüzde detaylı olarak sunulacak ve kullanıcılar için indirilebilir hale getirilecek.

İş Paketi 6: Test ve Optimizasyon (4-5 Ay)

Yazılımın birim ve entegrasyon testleri yapılacak, algoritmalar büyük dosyalar için optimize edilecek ve yazılım performansı iyileştirilecek. Tüm süreçlerin işleyişine yönelik test raporları hazırlanacak.

İş Paketi 7: Proje Sonlandırma ve Dokümantasyon (5 Ay)

Kullanım kılavuzu ve teknik dokümantasyon hazırlanacak, proje raporu oluşturulacak ve teslim süreci için gerekli hazırlıklar tamamlanacak. Proje çıktıları detaylı bir şekilde sunuma uygun hale getirilecek.

BÖLÜM 2: ÖNCEKİ ÇALIŞMALARIN ARAŞTIRILMASI

Bu bölümde, intihal tespiti alanında yaygın olarak kullanılan çeşitli araçlar incelenmektedir. Bu araçların özellikleri, avantajları, dezavantajları ve kullanım alanları detaylı bir şekilde ele alınarak, farklı ihtiyaçlara yönelik en uygun çözümlerin belirlenmesi amaçlanmaktadır. İnceleme hem ticari hem de açık kaynaklı araçları kapsamakta ve özellikle çoka çok analizler gibi karmaşık senaryolar için araçların yeterliliği değerlendirilmektedir.

2.1. Literatürdeki Benzer Çalışmalar

İntihal tespiti, akademik ve profesyonel dünyada önemli bir konudur. Bu alanda farklı amaçlar için geliştirilmiş çeşitli ticari ve açık kaynak araçlar mevcuttur. Özellikle çoka çok analizler ve grup ödevleri gibi karmaşık senaryolar için tespit aracının kapsamlı özelliklere sahip olması gerekir. Aşağıda yaygın kullanılan intihal tespit araçlarının detaylı incelemesi sunulmuştur.

2.1.1. iThenticate

iThenticate, bilimsel makaleler, profesyonel yayınlar ve yüksek kaliteli akademik içerikler için geliştirilen üst düzey bir intihal kontrol sistemidir. Yayınlanmış makaleler, konferans bildirileri ve bilimsel veri tabanları gibi kaynakları tarayarak, içeriklerin özgünlük durumunu yüksek bir doğrulukla analiz eder. Akademik yayınevleri ve araştırmacılar için özel çözümler sunan iThenticate, özellikle bilimsel makalelerin yayımlanmadan önceki orijinallik kontrollerinde kritik bir rol oynar. Kullanıcılarına sağladığı detaylı orijinallik raporları sayesinde, içeriklerin güvenilirliğini artırır ve intihal riskini minimize eder. iThenticate, tezler, bilimsel makaleler ve profesyonel yayınlar üzerinde çalışan araştırmacılar için ideal bir araçtır. Platformlar arası iş birliğini destekleyen yapısı, kullanıcı deneyimini güçlendirmektedir [3]. (https://www.ithenticate.com/)

Avantajları: Yüksek doğruluk, bilimsel yayın desteği, uluslararası standartlara uygun raporlama.

Dezavantajları: Yüksek maliyet, bireysel kullanıcılar için uygun değil, grup ödevleri ve çoka çok analizlerde yetersizlik.

Kullanım Alanları: Akademik yayınlar, bilimsel makaleler, yayınevleri.

2.1.2. Turnitin

Turnitin, dünya çapında akademisyenler ve eğitim kurumları tarafından tercih edilen, güvenilir bir intihal tespit platformudur. Geniş ve kapsamlı bir akademik veri tabanına sahip olan Turnitin, kullanıcıların içeriklerini kitaplar, makaleler, tezler ve diğer akademik çalışmalarla karşılaştırarak intihal oranını yüksek doğrulukla tespit edebilir. Turnitin'in sunduğu en önemli özelliklerden biri Orijinallik Raporudur. Bu rapor, belgelerde bulunan benzerlik oranlarını detaylı bir şekilde analiz ederek, içeriklerin özgünlük seviyesini değerlendirmeye yardımcı olur. Ayrıca, bu sistemin güçlü algoritmaları sayesinde, özellikle akademik standartların korunmasına büyük katkı sağlanır. Üniversiteler, akademisyenler ve araştırmacılar için vazgeçilmez bir araç olan Turnitin, aynı zamanda akademik dürüstlük ve etik çalışmaların desteklenmesine de önemli bir rol oynamaktadır [2], [3]. (https://tr.turnitin.com/)

Avantajları: Geniş veri tabanı, detaylı raporlama, kullanıcı dostu arayüz.

Dezavantajları: Yüksek maliyet, çoka çok analiz eksikliği.

Kullanım Alanları: Akademik kurumlar, üniversiteler, okullar.

2.1.3. PlagScan

PlagScan, daha düşük maliyetli ve açık kaynak bir intihal tespit aracıdır. Belirli bir düzeye kadar etkili bir çözüm sunmasına rağmen, gelişmiş analiz gereksinimlerini karşılamada yetersizdir [3]. (https://www.plagscan.com/)

Avantajları: Düşük maliyetli lisans modeli, çevrimdışı belgelerle karşılaştırma.

Dezavantajları: Çoka çok analiz eksikliği, veri tabanı kısıtlamaları.

Kullanım Alanları: Eğitim kurumları, küçük işletmeler, bireysel kullanıcılar.

2.1.4. İntihal.net

İntihal.net, Türkçe akademik çalışmalara odaklanmış, yerli bir intihal tespit aracıdır. Türk üniversiteleri, akademisyenler ve araştırmacılar tarafından yaygın bir şekilde kullanılan bu araç, Türkçe içeriklerin özgünlük kontrollerinde oldukça başarılıdır. İntihal.net, akademik ve bilimsel raporlamalar için kapsamlı bir tarama sistemi sunar ve kullanıcı dostu arayüzü sayesinde kolay bir kullanım deneyimi sağlar. Bu platform, özellikle Türkçe tezler, makaleler ve diğer akademik çalışmalar üzerinde çalışan araştırmacılar için idealdir. Türkçe kaynaklara erişim sağlama konusundaki başarısı, İntihal.net'i yerli akademik çalışmalar için vazgeçilmez bir araç haline getirmiştir. Ayrıca, kullanıcılara detaylı raporlar sunarak, içeriklerin intihal oranını doğru bir şekilde analiz eder ve araştırma süreçlerinin verimliliğini artırır. (https://intihal.net/)

Avantajları: Türkçe dil desteği, kullanıcı dostu arayüz, Türkçe kaynaklara erişim ve detaylı raporlama.

Dezavantajları: Dil desteği sınırlaması (sadece Türkçe), veri tabanı kapsamının darlığı (uluslararası kaynaklara sınırlı erişim), çoka çok analiz eksikliği

Kullanım Alanları: Türkiye'deki akademik kurumlar, Türkçe içerik üretenler.

2.1.5. PlagiarismChecker

PlagiarismChecker, tamamen ücretsiz olması ve kullanımı kolay arayüzüyle bireysel kullanıcılar için pratik bir intihal tespit aracıdır. Özellikle kısa metinlerin hızlı bir şekilde analiz edilmesine olanak tanıyan bu araç, web tabanlı çalıştığı için herhangi bir kurulum gerektirmez. Kopyala yapıştır yöntemiyle çalışan PlagiarismChecker, metinlerin temel düzeyde taranmasını sağlar ve kullanıcılarına kısa sürede sonuç sunar. Sağladığı temel raporlarla, kullanıcıların içeriklerinin özgünlük durumunu anlamalarına yardımcı olur. Blog yazıları, kısa makaleler ve ödevler gibi basit içeriklerde etkili bir şekilde çalışan bu araç, karmaşık akademik metinler yerine daha

küçük ve bireysel çalışmalar için idealdir. Özellikle öğrenciler ve bireysel yazarlar arasında popülerdir [4]. (https://plagiarismdetector.net/)

Avantajları: Tamamen ücretsiz, kullanımı kolay arayüz, hızlı analiz, kurulum gerektirmemesi.

Dezavantajları: Temel düzeyde tarama, karmaşık analizlerde yetersizlik, daha az kapsamlı veri tabanı.

Kullanım Alanları: Bireysel kullanıcılar, öğrenciler, blog yazarları, kısa metinler.

Tablo 2.1.'da çalışmaların birbirleriyle karşılaştırıldığı tablo verilmiştir.

Tablo 2.1. Araç Karşılaştırma Tablosu

Araç Adı	Avantajlar	Dezavantajlar	Kullanım Alanları	Ücret Durumu
iThenticate	Geniş veri tabanı, detaylı raporlar, kullanıcı dostu arayüz	Yüksek maliyet, çoka çok eksikliği	Akademik kurumlar, üniversiteler, okullar	Kurumsal fiyatlandırma
Turnitin	Yüksek doğruluk, bilimsel yayın desteği, uluslararası standartlara uygun raporlama	Yüksek maliyet, bireysel için uygun değil, grup ödevleri ve çoka çok analizlerde yetersizlik	Akademik yayınlar, bilimsel makaleler, yayınevleri	Kurumsal fiyatlandırma
PlagScan	Düşük maliyet, çevrimdışı destek	Çoka çok eksikliği, sınırlı veri tabanı	Eğitim kurumları, küçük işletmeler, bireysel kullanıcılar	Bireysel ve kurumsal fiyatlandırma
İntihal.net	Türkçe dil desteği, kullanıcı dostu arayüz,	Dil desteği sınırlaması (sadece Türkçe),	Türkiye'deki akademik kurumlar, Türkçe	Kurumsal fiyatlandırma

	Türkçe kaynaklara erişim ve detaylı raporlama	veri tabanı kapsamının darlığı (uluslararası kaynaklara sınırlı erişim), çoka çok analiz eksikliği	içerik üretenler	
PlagiarismChecker	Tamamen ücretsiz, kullanımı kolay arayüz, hızlı analiz, kurulum gerektirmemesi	Temel düzeyde tarama, karmaşık analizlerde yetersizlik, daha az kapsamlı veri tabanı	Bireysel kullanıcılar, öğrenciler, blog yazarları, kısa metinler	Ücretsiz

2.2. Mevcut Çalışmaların Karşılaştırılması ve Eksikleri

Mevcut araçlar, bire bir belge karşılaştırmasında etkili çözümler sunarken, belgelerin birbirleriyle çoka çok eşleşme yöntemiyle analiz edilmesi gibi daha karmaşık ihtiyaçları karşılamakta yetersiz kalmaktadır. Özellikle sınıf düzeyinde yapılan grup ödevlerinde ya da öğrencilerin aynı konu üzerinde çalıştığı bireysel ödevlerde, belgeler arasındaki benzerliğin kapsamlı bir şekilde analiz edilmesi gerekmektedir. Ancak, mevcut araçların çoğu bu tür gereksinimlere odaklanmamıştır [3].

BÖLÜM 3: YÖNTEM VE TASARIM

Bu bölümde, projenin tasarım ve geliştirme süreci, gereksinim analizinden sistem mimarisine, metin analizi yöntemlerinden test süreçlerine kadar tüm detaylarıyla anlatılmaktadır. Riskler, çözümleri ve beklenen faydalar da ele alınmıştır.

3.1. Tasarım İhtiyaçları

Projenin başarılı bir şekilde hayata geçirilmesi için çeşitli tasarım ihtiyaçları dikkate alınmıştır. Bu ihtiyaçlar, kullanıcı deneyimini, teknolojik altyapıyı, maliyet etkinliğini, sistem entegrasyonunu ve uyumluluğunu kapsamaktadır.

3.1.1. Kullanıcı ihtiyaç analizi

Bu sistemin temel amacı, ödev, proje ve diğer akademik içeriklerin özgünlüğünü analiz etmektir. Bu nedenle, kullanıcı odaklı bir tasarım yaklaşımı benimsenmiştir. Kullanıcı ihtiyaç analizi sonucunda aşağıdaki gereksinimler belirlenmiştir.

Kullanıcı Dostu Arayüz: Sistem, her seviyedeki kullanıcının kolayca anlayabileceği ve kullanabileceği sezgisel bir arayüze sahip olmalıdır. Karmaşık menülerden ve teknik terimlerden kaçınılarak, basit ve anlaşılır bir dil kullanılmalıdır.

Belge Yükleme ve Yönetimi: Kullanıcılar, farklı formatlardaki belgelerini (örneğin, .pdf, .doc, .docx) kolayca yükleyebilmeli ve yönetebilmelidir. Yükleme süreci hızlı ve güvenli olmalıdır.

Karşılaştırma Başlatma: Kullanıcılar, yükledikleri belgeler arasında karşılaştırma işlemini basit bir şekilde başlatabilmelidir. Gerekirse, karşılaştırma parametrelerini (örneğin, benzerlik eşiği) ayarlayabilmelidir.

Detaylı Raporlar: Sistem, analiz sonuçlarını detaylı ve anlaşılır raporlar halinde sunmalıdır. Raporlar, benzerlik oranlarını, eşleşen metin bölümlerini ve diğer ilgili bilgileri içermelidir. Görselleştirmeler (grafikler, tablolar) raporun anlaşılırlığını artırmalıdır.

Görselleştirme: Analiz sonuçları, grafikler ve tablolar aracılığıyla görselleştirilerek, kullanıcıların veriyi daha kolay yorumlaması sağlanmalıdır. Benzerlik oranları, eşleşen metin bölümleri ve diğer önemli veriler görsel olarak sunulmalıdır.

Hızlı Geri Bildirim: Sistem, analiz sonuçlarını mümkün olan en kısa sürede vermelidir. Uzun süren analizler için kullanıcıya ilerleme durumu hakkında bilgi verilmelidir.

Bu genel ihtiyaçların sistemdeki karşılıkları ve ilgili kullanıcı rolleri aşağıdaki Şekil 3.1.'da gösterilmiştir.

EYLEM MADDESİ	ERİŞİM SAĞLAYAN KİŞİ	İLGİLİ İHTİYAÇ
Sistemde bir içerik (ödev, proje vb.) oluşturur.	Öğretim Görevlisi	İçerik Oluşturma ve Yönetme
Bir içeriğe dosya yükler.	Öğrenci	Dosya Yükleme
Kendi oluşturduğu bir içerikteki dosyaların karşılaştırılması emrini verir.	Öğretim Görevlisi	Özgünlük Kontrolü, Karşılaştırma Başlatma
Karşılaştırma işlemi sonrası kendi oluşturduğu içeriklerdeki detaylı inceleme raporlarını görür.	Öğretim Görevlisi	Detaylı Raporlar, Görselleştirme, Hızlı Geri Bildirim
Karşılaştırma işlemi sonrası yalnızca kendi dosyasıyla ilgili detaylı inceleme raporunu görür.	Öğrenci	Detaylı Raporlar, Görselleştirme, Hızlı Geri Bildirim

Şekil 3.1. Kullanıcı İhtiyaç Analizi

3.1.2. Teknoloji gereksinimleri

Bu sistemin geliştirilmesinde, performans, ölçeklenebilirlik, güvenlik ve kullanıcı

deneyimi gibi faktörler göz önünde bulundurularak bir dizi teknoloji seçilmiştir. Kullanılan temel teknolojiler ve seçim nedenleri aşağıda detaylandırılmıştır:

Ön Yüz (Frontend): Vue.js

Kullanıcı etkileşimini yönetmek ve dinamik bir arayüz oluşturmak için JavaScript tabanlı bir framework olan Vue.js tercih edilmiştir.

Arka Yüz (Backend): Flask (Python)

API uç noktalarını oluşturmak, veri işleme süreçlerini yönetmek ve veri tabanı ile iletişim kurmak için Python tabanlı hafif bir web framework'ü olan Flask kullanılmıştır.

Veri Tabanı: MSSQL

Yüklenen belgelerin, analiz sonuçlarının, kullanıcı bilgilerinin ve diğer verilerin güvenli ve verimli bir şekilde depolanması ve yönetilmesi için Microsoft SQL Server (MSSQL) veri tabanı kullanılmıştır.

API Testi: Postman

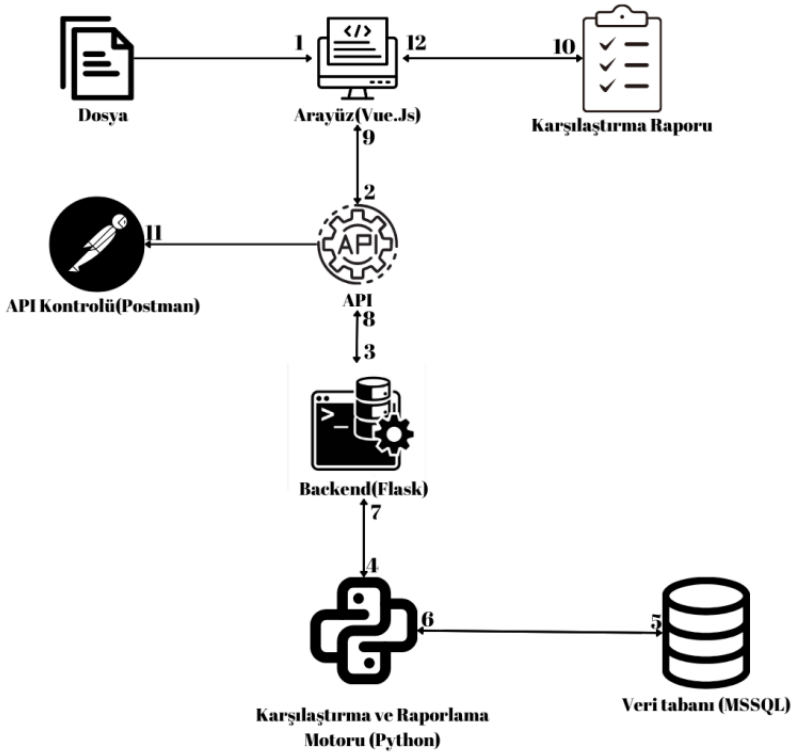
API uç noktalarının doğru çalıştığından, beklenen yanıtları verdiğinden ve performans gereksinimlerini karşıladığından emin olmak için Postman kullanılmıştır.

Programlama Dili: Python

Metin karşılaştırma algoritmalarının geliştirilmesi, veri analizi ve sonuçların görselleştirilmesi için Python programlama dili kullanılmıştır.

Sistemin Ana Hatları:

Aşağıda verilen Şekil 3.2.'da istem mimarisinin genel görünümü. Bu diyagram, sistem bileşenlerinin (ön yüz, arka yüz, veri tabanı, vb.) ve aralarındaki etkileşimin nasıl olduğunu göstermektedir.



Şekil 3.2. Sistem Mimarisi

3.1.3. Maliyet gereksinimleri

Projenin maliyet etkinliği önemli bir öncelik olarak belirlenmiş ve bu doğrultuda mümkün olduğunca açık kaynak teknolojiler tercih edilmiştir. Flask (arka yüz için), Vue.js (ön yüz için), Python ve ilgili kütüphaneler (metin analizi için) gibi temel bileşenler açık kaynaklı ve ücretsizdir. Ancak, MSSQL veri tabanı için uygun bir lisanslama bütçesi ayrılması gerekmektedir. MSSQL'in farklı sürümleri ve lisanslama seçenekleri bulunmaktadır.

3.1.4. Entegrasyon ve uyum gereksinimleri

Arayüz ile arka yüz iletişimi API'lerin aracılığıyla sağlanacaktır. Postman ile test edilen API'lerin entegrasyonu, Flask arka yüz yapısına ve Vue.js ön yüz yapısına uygun şekilde yapılacaktır.

3.2. Tasarım İçin Kullanılacak Sistem ve Alt Sistemler

3.2.1. Ön Yüz (Frontend): Vue.js

Vue.js'in bileşen tabanlı yapısı, kodun yeniden kullanılabilirliğini artırır ve geliştirme sürecini hızlandırır. Ayrıca, performansı yüksektir ve zengin bir ekosisteme sahiptir. Tek sayfa uygulamaları (SPA) geliştirmek için idealdir ve kullanıcı deneyimini önemli ölçüde iyileştirir. Bu proje özelinde, dosya yükleme, analiz sonuçlarının görselleştirilmesi ve kullanıcı etkileşimlerinin yönetimi gibi modüller Vue.js ile geliştirilecektir. Aşağıda Şekil 3.3.'da yönlendirme mekanizmasından bir görsel verilmiştir.

3.2.2. Arka Yüz (Backend): Flask (Python)

Flask'ın minimalist yapısı, yüksek performans gerektiren uygulamalar için uygundur. Esnek ve özelleştirilebilir yapısı sayesinde, projenin özel ihtiyaçlarına göre kolayca adapte edilebilir. Python'un güçlü kütüphaneleriyle (örneğin, metin işleme ve veri analizi için) entegrasyonu kolaydır. Bu proje için, dosya yükleme, analiz başlatma, sonuçları veri tabanından alma ve ön yüze gönderme gibi işlemler Flask ile yönetilecektir.

3.2.3. Veri Tabanı: MSSQL

MSSQL, bu proje kapsamında yüksek performanslı veri depolama ve işleme ihtiyaçlarını karşılamak için seçilmiştir. Veri tabanı, yüklenen belgeleri, karşılaştırma sonuçlarını, kullanıcı erişim bilgilerini ve görselleştirme için gerekli istatistikleri depolayacaktır. Tablolar normalize edilerek veri tutarlılığı sağlanacak ve karmaşık sorgular için indeksleme yapılacaktır.

3.2.4. Postman ile API Testi

Postman, API geliştirme ve test süreçlerini kolaylaştıran güçlü bir araçtır. API istekleri gönderme, yanıtları inceleme, test senaryoları oluşturma ve otomatik testler

çalıştırma gibi özellikleri sunar. Bu proje için, farklı senaryolar altında API'lerin doğruluğunu, performansını ve güvenliğini test etmek için Postman kullanılacaktır.

3.2.5. Python ile Veri İşleme

Python, zengin kütüphane ekosistemi (NLTK, spaCy, scikit-learn, matplotlib, seaborn, plotly vb.) sayesinde metin işleme, doğal dil işleme (NLP), makine öğrenimi ve veri görselleştirme gibi alanlarda yaygın olarak kullanılan bir dildir.

3.2.6. Benzerlik Analizi Algoritmalarının Çalışma Prensipleri ve Kullanım Senaryoları

Metin benzerliği analizi, doğal dil işleme (NLP) ve metin madenciliği alanlarında önemli bir yer tutmaktadır. Farklı metinler arasındaki benzerlikleri tespit etmek, metin sınıflandırma, intihal tespiti, arama motoru optimizasyonu ve içerik öneri sistemleri gibi birçok uygulama için gereklidir. Aşağıda, en yaygın kullanılan metin benzerliği algoritmalarının çalışma prensipleri ve kullanım senaryoları açıklanmaktadır.

Cosine Similarity: Cosine benzerliği, metinleri vektörler olarak temsil edip, bu vektörler arasındaki açıyı (cosine) hesaplayarak benzerliği ölçen bir yöntemdir. Vektörler, metnin içerdiği kelimelerin sıklıkları ile temsil edilir. Cosine benzerliği, iki metnin yönlerinin birbirine ne kadar yakın olduğunu (veya paralel olduğunu) ölçer ve bu benzerliği -1 ile 1 arasında bir değerle ifade eder. Cosine Similarity, belge benzerliği ve doküman sıralama gibi uygulamalar için son derece etkilidir. E-posta sınıflandırma, web arama motorları, öneri sistemleri ve metin madenciliği gibi uygulamalarda büyük veri ile çalışırken yüksek verimlilik sağlar. Ayrıca, uzun metinler ve anlamlı benzerlikler üzerine odaklanmak gerektiğinde daha doğru sonuçlar verir. Yüksek boyutlu veri setlerinde de başarılı performans sergileyen bir algoritmadır [5].

Monge-Elkan: Monge-Elkan algoritması, metin benzerliklerini ölçmek için kullanılan güçlü ve esnek bir yöntemdir. Bu algoritma, özellikle farklı şekillerde ifade edilen veya yazım hataları içeren kısa metinlerin karşılaştırılmasında etkili sonuçlar verir. Monge-Elkan, iki metindeki her bir kelimeyi diğer metindeki tüm kelimelerle eşleştirir ve bu eşleşmelerin benzerlik skorlarını hesaplar. Bu süreçte genellikle başka bir string benzerlik algoritması (örneğin, Levenshtein mesafesi veya Jaro-Winkler) alt algoritma olarak kullanılır. Her bir kelime için en yüksek benzerlik skoru seçilir ve tüm skorların ortalaması alınarak iki metin arasındaki genel benzerlik değeri elde edilir. Bu algoritmanın en belirgin avantajı, küçük yazım hatalarına veya eşanlamlı ifadelerin varlığına karşı toleranslı olmasıdır. Monge-Elkan, birebir eşleşme aramak yerine en yüksek benzerliği seçerek metinler arasındaki anlam bütünlüğünü korur. Özellikle, müşteri veritabanı eşleştirme, adres karşılaştırma ve kısa metinli doğal dil işleme gibi uygulamalarda yaygın olarak kullanılır. Bununla birlikte, Monge-Elkan algoritmasının kullanımında dikkat edilmesi gereken bazı noktalar vardır. Kullanılan alt algoritmanın seçimi, sonuçların doğruluğu üzerinde doğrudan etkilidir. Örneğin, Levenshtein mesafesi yazım hatalarına karşı hassasken, Jaro-Winkler algoritması daha kısa metinler üzerinde etkili çalışabilir. Bu nedenle, Monge-Elkan'ın en uygun sonuçları vermesi için veri türüne ve karşılaştırma senaryosuna uygun bir alt algoritma seçimi yapılmalıdır [6].

Jaro-Winkler Similarity: Jaro-Winkler benzerliği, özellikle kısa metinler ve adlar arasındaki benzerliği ölçmek için tasarlanmış bir algoritmadır. Bu algoritma, metinlerdeki karakter benzerliklerini değerlendirir ve küçük yazım hatalarına duyarlıdır. Jaro-Winkler, kelimelerdeki karakter sırasındaki benzerliği ve yer değiştirmeleri dikkate alarak bir skor üretir. Bu algoritma, daha uzun metinlerdeki benzerlikleri doğru tespit etmekte zorlanabilir, ancak küçük farkları oldukça hassas bir şekilde belirleyebilir. Jaro-Winkler, özellikle ad eşleştirme, yazım hatası düzeltme, telefon numarası ve adres doğrulama gibi uygulamalar için etkilidir. Veri doğrulama, spam filtreleme ve benzer adların tespiti gibi işlemler için yaygın olarak tercih edilen bir algoritmadır [7].

Sorensen-Dice Coefficient: Sorensen-Dice katsayısı, iki metin arasındaki benzerliği ölçmek için kullanılan bir diğer küme benzerlik algoritmasıdır. Bu katsayı, her iki

metnin ortak kelimelerinin sayısının, her iki metnin toplam kelimeleri ile karşılaştırılmasıyla hesaplanır. Sorensen-Dice katsayısı, özellikle metinlerin ortak ögelerinin oranını belirlemek için etkilidir. Sorensen-Dice Coefficient, özellikle kısa metinlerde ve kelime kümesi benzerliği analizi için kullanılır. Metinlerin karşılaştırılması, belge öneri sistemleri ve etiketleme sistemlerinde başarılı sonuçlar elde etmek için uygundur. Ayrıca, metinler arasındaki küçük benzerlikleri tespit etmekte oldukça etkili bir algoritmadır. Her algoritma, farklı veri yapıları, metin türleri ve kullanım senaryolarına göre değişik performanslar sergileyebilir. Bu nedenle, doğru algoritmayı seçmek, belirli bir kullanım durumu için doğru sonuca ulaşabilmek adına son derece önemlidir. Örneğin, TF-IDF genellikle büyük veri setlerinde metinlerin içerik benzerliğini analiz etmek için mükemmel bir seçimken, Levenshtein mesafesi daha çok yazım hatası tespiti için etkili olur. Cosine Similarity, uzun metinlerdeki anlamlı benzerlikleri tespit etmek için ideal bir yöntemken, Jaro-Winkler algoritması kısa metinlerdeki yazım hatalarını doğru şekilde belirleyebilir [8].

Jaccard Index: Jaccard benzerliği, iki küme arasındaki ortak öğelerin oranını ölçer. Bu algoritma, metinleri kelime kümeleri olarak kabul eder ve iki küme arasındaki benzerliği, küme kesişimi ile birleşiminin oranı olarak hesaplar. Jaccard Index, genellikle kelimeler arasındaki ortaklıkları, metinler arasındaki yapısal benzerlikleri değerlendiren bir metottur. Jaccard Index, özellikle kısa metinler ve kelime kümesi analizleri için oldukça etkili bir algoritmadır. Metin madenciliği ve veritabanı sorgulama gibi alanlarda, etiketleme, doküman eşleştirme ve kümelenme (clustering) gibi işlemler için yaygın bir şekilde kullanılır. Jaccard, özellikle sosyal medya analizleri, hashtag analizleri ve metin analizi gibi ortamlarda, metinlerin içerik benzerliklerini daha iyi anlamak için faydalıdır [9].

TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF, bir kelimenin, belirli bir metin veya belge içinde ne kadar önemli olduğunu belirlemek için kullanılan istatistiksel bir yöntemdir. İki temel bileşenden oluşur Term Frequency (TF), kelimenin bir metin içindeki sıklığını ölçerken; Inverse Document Frequency (IDF), kelimenin tüm belgelerde ne kadar yaygın olduğunu değerlendiren bir parametredir. Bu iki bileşen birleştirilerek, her kelimenin metin içindeki gerçek önem

derecesi hesaplanır. Yüksek bir TF-IDF değeri, kelimenin sadece belirli bir metinde sıkça geçtiğini değil, aynı zamanda o metinde özgün ve anlamlı bir yere sahip olduğunu gösterir.

TF-IDF, özellikle büyük veri setlerinde, metin madenciliği ve içerik analizi gibi uygulamalarda yaygın olarak kullanılır. Arama motorları, doküman sınıflandırma ve belge öneri sistemleri gibi alanlarda oldukça etkilidir çünkü bu yöntem, metinlerin içeriğine dair önemli kelimeleri çıkarmada oldukça başarılıdır. Ayrıca, bu algoritma, sık kullanılan kelimelerin veya durak kelimelerin (stop words) metin analizi üzerinde olumsuz bir etkisi olmasını engeller [10].

Şekil 3.3.'da TF-IDF formülü görselleştirilmiştir.

Şekil 3.3. TF-IDF Formülü

Levenshtein Distance: Levenshtein Distance, iki metin arasında yapılan değişiklikleri (ekleme, silme, değiştirme) ölçer. Bu yöntem, metinlerin harf düzeyinde ne kadar farklı olduğunu değerlendirmek için kullanılabilir [11]. Aşağıdaki, Denklem 3.1. ve Şekil 3.4.'da formül ve örnek verilmiştir.

$$\mathrm{lev}_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{eğer } \min(i,j) = 0, \\ min \begin{cases} \mathrm{lev}_{a,b}(i-1,j)+1 \\ \mathrm{lev}_{a,b}(i,j-1)+1 \\ \mathrm{lev}_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{diğer.} \end{cases}$$

Denklem 3.1. Levenshtein Distance Formülü

	-	Ş	A	N	L	I	U	R	F	A
-	0	1	2	3	4	5	6	7	8	9
S	1	0	1	2	3	4	5	6	7	8
A	2	1	0	1	2	3	4	5	6	7
K	3	2	1	1	2	3	4	5	6	7
A	4	3	2	2	2	3	4	5	6	7
R	5	4	3	3	3	3	4	4	5	6
Y	6	5	4	4	4	4	4	5	6	7
A	7	6	5	5	5	5	5	6	7	6

Şekil 3.4. Levenshtein Distance Örneği

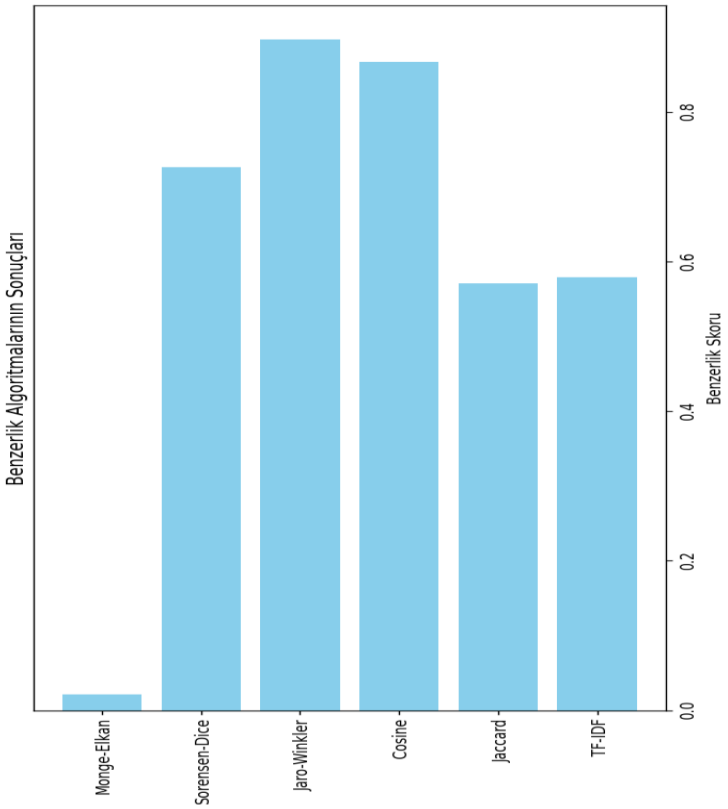
Algoritmaların Karşılaştırılması

Farklı metin benzerliği algoritmalarının performansını karşılaştırmak, hangi algoritmanın belirli bir uygulama için en uygun olduğunu belirlemek açısından büyük önem taşır. Bu bölümde, TF-IDF, Jaccard Index, Cosine Similarity, Jaro-Winkler Similarity ve Sorensen-Dice Coefficient algoritmalarının metinler üzerinde benzerlik oranı tespit edilmiştir. Ayrıca, test aşamasında farklı metinler kullanılarak hangi algoritmaların hangi tür metin verileri ve kullanım senaryoları için daha etkili olduğunu tespit edilmiştir. Bu test aşamasında kendi programımız kullanılmış ve sonuçlar aşağıda verilmiştir.

Test 1: Şekil 3.5.' da verilen test girdisi sonuçları Şekil 3.6.'da gösterilmiştir.

```
text1 = """
Web gelistirme modern cagda cok onemlidir."""
text2 = """
Web gelistirme gunumuzde cok onemlidir."""
```

Şekil 3.5. Test 1 Girdisi

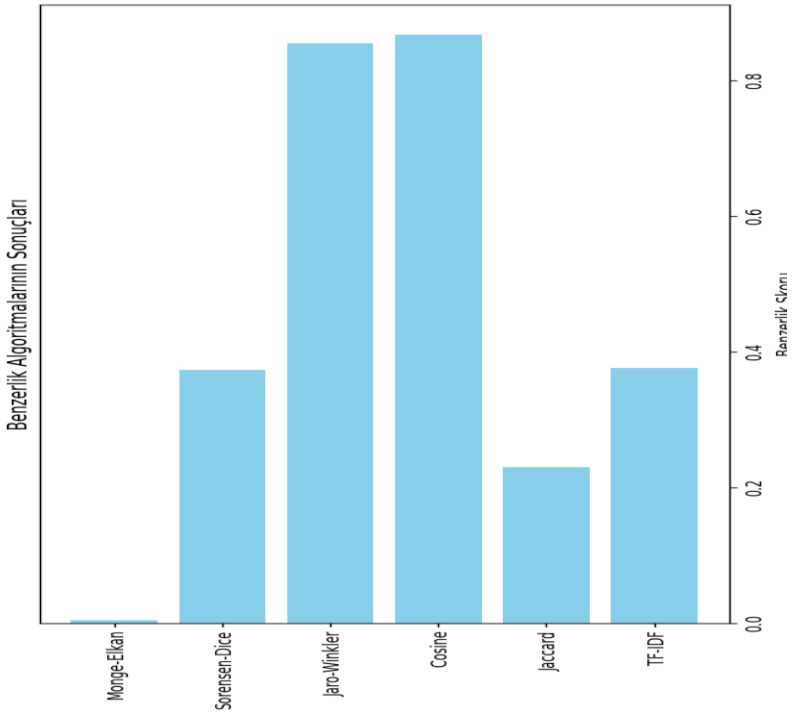


Şekil 3.6. Test 1 Sonucu

Test 2: Şekil 3.7.' da verilen test girdisi sonuçları Şekil 3.8.'da gösterilmiştir.

```
text1 = """
Python, gelismis bir programlama dilidir. Hem basit hem de guclu bir yapisi vardir.
Yazilim dunyasinda genis bir kullanima sahiptir ve bircok farkli amac icin kullanilabilir"""
text2 = """
Python, guclu ve cok yonlu bir programlama dilidir. Kullanimi kolaydir ve yazilim projelerinde siklikla tercih edilir.
Farkli alanlarda kullanilabilecek genis bir kapasiteye sahiptir"""
```

Şekil 3.7. Test 2 Girdisi

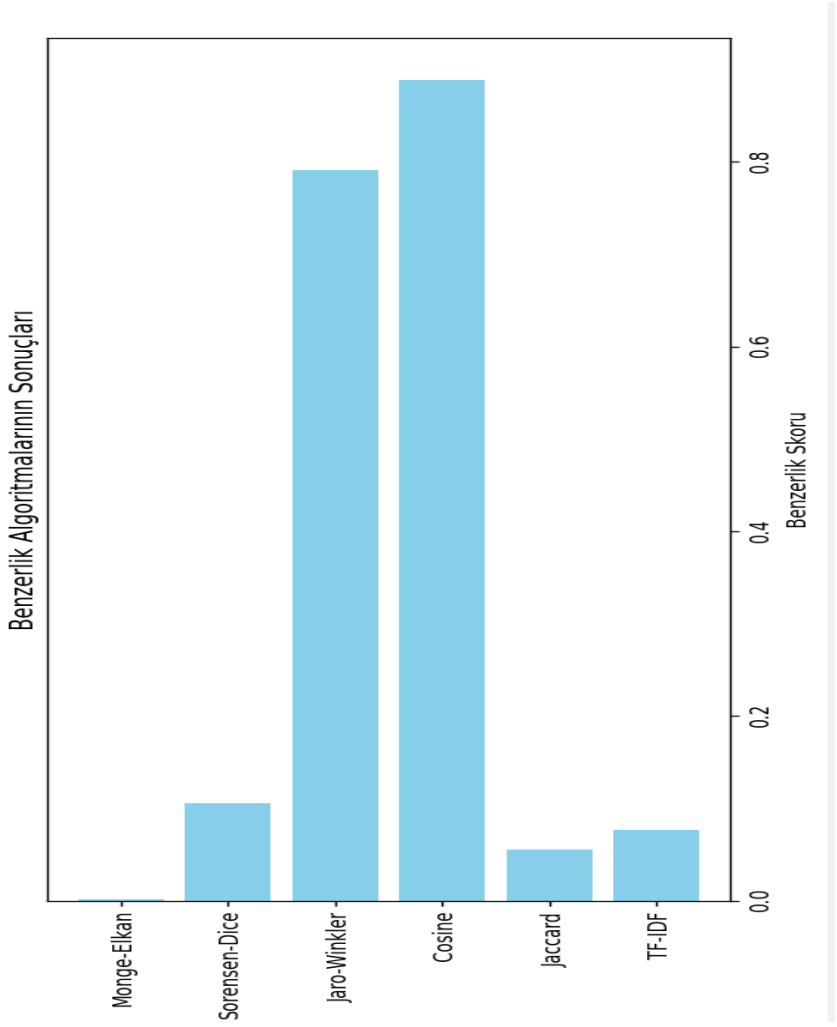


Şekil 3.8. Test 2 Sonucu

Test 3: Şekil 3.9.' da verilen test girdisi sonuçları Şekil 3.10.'da gösterilmiştir.

```
text1 = """
Web gelistirme sureci, her gecen gun daha fazla onem kazanmaktadir. Kullanicilarin beklentilerini karsilayacak
hizli ve kullanci dostu siteler gelistirmek,gunumuzun temel gerekliliklerinden biridir.
Bu nedenle, HTML, CSS ve JavaScript gibi dillerin yani sira, dinamik web sayfalari icin framework kullanimi oldukca yayginlasmistir.
Frameworkler, gelistiricilerin islerini kolaylastirarak daha verimli calismalarini saglar."""
text2 = """
Mobil cihazlar, artik hayatimizin ayrilmaz bir parcasi haline gelmistir. Android ve iOS isletim sistemleri icin mobil uygulamalar
gelistirilirken, uygulamanin hizi, verimliligi ve kullanci deneyimi on plana cikmaktadir. Gelistiriciler, mobil uygulamalar icin özellikle
React Native veya Flutter gibi frameworkleri tercih ederken,
platformlar arasi uyumluluga dikkat etmektedirler"""
```

Şekil 3.9. Test 3 Girdisi



Şekil 3.10. Test 3 Sonucu

3.3. Yöntem

Bu bölümde, sistemin geliştirilmesinde izlenen yöntemler, kullanılan teknolojiler ve uygulanan süreçler detaylandırılmaktadır. Sistem, modüler bir yaklaşımla, ön yüz, arka yüz, veri işleme ve veri tabanı katmanları olarak tasarlanmıştır.

3.3.1. Flask Tabanlı Arka Yüz Geliştirme

Sistemin arka yüzü, Python tabanlı hafif bir web framework'ü olan Flask kullanılarak geliştirilmiştir. Flask'ın esnek ve minimalist yapısı, RESTful API'ler oluşturmak için uygun bir zemin sağlamıştır. Arka yüz geliştirme sürecinden aşağıdaki örnekler verilmiştir:

API Uç Noktaları: Aşağıda bazı API uç noktalarının görevleri aktarılmıştır:

POST /upload: Kullanıcıların dosyalarını sisteme yüklemesini sağlayan uç noktadır. Yüklenen dosyalar sunucuda güvenli bir dizine kaydedilir ve dosya meta verileri (ad, boyut, yüklenme zamanı vb.) MSSQL veri tabanına kaydedilir.

POST /compare: Karşılaştırma işlemini başlatır. Bu uç nokta, gerekli analiz parametrelerini alır ve analiz işlemini gerçekleştirmek üzere Python modülüne yönlendirir.

GET /results/<analysis_id>: Belirli bir analiz ID'sine ait sonuçları döndüren uç noktadır. Sonuçlar, veri tabanından alınarak uygun bir formatta (JSON gibi) ön yüze iletilir.

Aşağıda Şekil 3.11.'da Flask uygulamasından uç nokta(endpoint) örnekleri verilmiştir.

Şekil 3.11. Flask Endpoint

Güvenlik: Sistem güvenliği ön planda tutularak aşağıdaki önlemler alınmıştır:

Dosya yükleme sırasında yalnızca izin verilen dosya türleri (.pdf, .doc, .docx) kabul edilir. Bu, olası güvenlik açıklarını en aza indirir.

Kullanıcı kimlik doğrulaması için JWT tabanlı bir mekanizma kullanılmıştır. Bu, güvenli ve ölçeklenebilir bir kimlik doğrulama yöntemi sağlar.

Performans: Yüksek performans sağlamak amacıyla aşağıdaki optimizasyonlar yapılmıştır:

Veritabanı işlemlerini optimize etmek için Flask-SQLAlchemy ORM kullanılmıştır. Bu, veritabanı etkileşimlerini daha verimli hale getirir.

3.3.2. Python ile Metin Analizi

Metin tabanlı analiz işlemleri için Python ve ilgili kütüphaneler kullanılmıştır. Analiz süreci şu adımları içerir:

Ön İşleme: Metin analizi için verilerin hazırlanması amacıyla aşağıdaki işlemler gerçekleştirilir:

Belgelerden metin çıkarma işlemi için PyPDF2 ve python-docx kütüphaneleri kullanılmıştır.

Çıkarılan metinler, gereksiz kısımların (kapak sayfası, özet, içindekiler vb.) filtrelenmesi için önceden tanımlanmış anahtar kelimelerle analiz edilir.

Doğal dil işleme (NLP) teknikleri uygulanarak NLTK ve spaCy kütüphaneleri ile durak kelimeler (stop words) temizlenir ve metin tokenizasyonu yapılır.

Benzerlik Analizi: Belgeler arasındaki benzerliği ölçmek için TF-IDF (Term Frequency-Inverse Document Frequency), Levenshtein Distance, Jaccard Index ve Cosine Similarity algoritmaları kullanılmıştır.

Bu algoritmalar, belgelerin farklı yönlerden benzerliğini değerlendirmek için birlikte veya ayrı ayrı kullanılabilir. Elde edilen sonuçlar, eşleşen bölümlerin satır ve paragraf bazında detaylı analizini içeren bir yapıda raporlanır.

Sonuçların Raporlanması: Analiz sonuçları, Python veri görselleştirme kütüphaneleri (matplotlib, seaborn, plotly) kullanılarak anlamlı ve okunabilir raporlar haline getirilir ve veri tabanında saklanır. Bu sonuçlar, Flask API aracılığıyla ön yüze iletilerek kullanıcılara sunulur.

3.3.3. MSSQL Tabanlı Veri Depolama

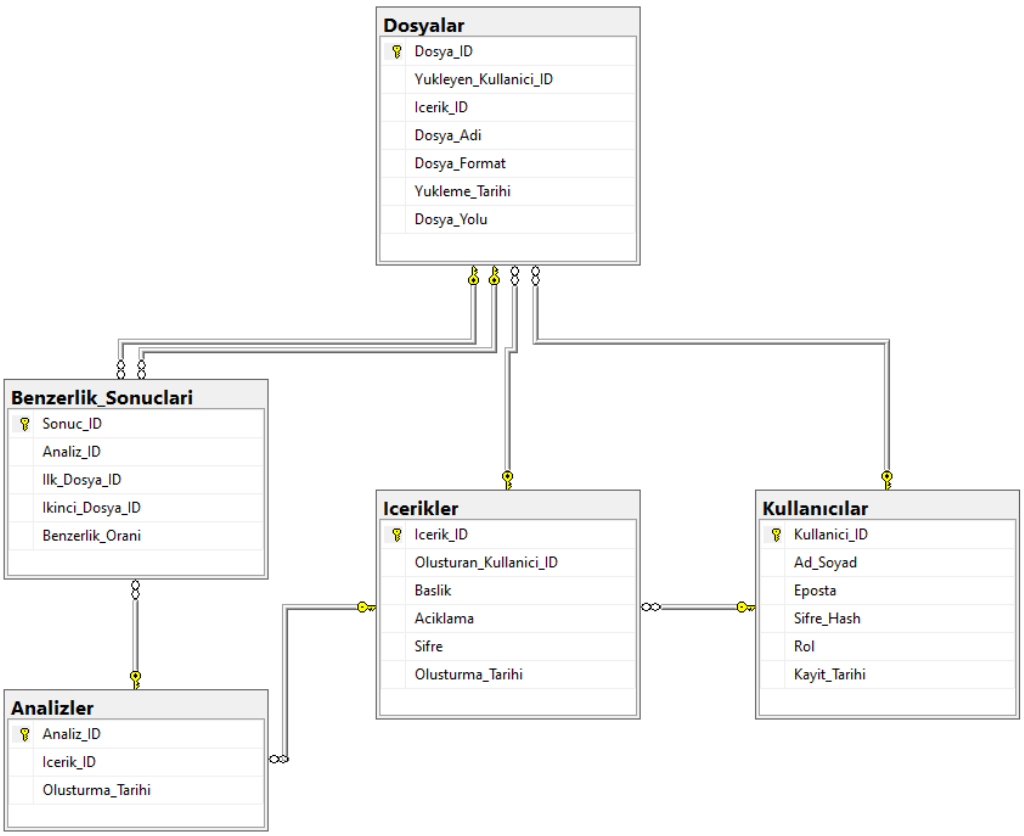
Verilerin kalıcı olarak saklanması ve etkin bir şekilde yönetilmesi için Microsoft SQL Server (MSSQL) veritabanı kullanılmıştır. Veritabanı şeması aşağıdaki tabloları içerir:

Kullanıcılar: Sistem kullanıcılarının kimlik bilgileri ve yetkilendirme bilgileri bu tabloda saklanır.

Yüklenen Dosyalar: Yüklenen dosyaların meta verileri (dosya adı, boyutu, yükleme tarihi, kullanıcı ID'si vb.) bu tabloda tutulur.

Analiz Sonuçları: Her analiz işlemine ait benzerlik oranları, eşleşen bölümler ve diğer ilgili veriler, benzersiz bir analiz ID'si ile ilişkilendirilerek bu tabloda saklanır.

Şekil 3.12.'da veri tabanı diyagramı gösterilmiştir.



Şekil 3.12. Veri Tabanı Tablosu

Veritabanı performansını optimize etmek için aşağıdaki teknikler kullanılmıştır:

Sık kullanılan sorgularda performansı artırmak için uygun indeksler oluşturulmuştur.

Karmaşık sorguların performansını artırmak ve veritabanı sunucusundaki yükü azaltmak için saklı yordamlar (stored procedures) kullanılmıştır.

3.3.4. Vue.js ile Frontend Geliştirme

Kullanıcı dostu ve etkileşimli bir arayüz oluşturmak için Vue.js framework'ü kullanılmıştır. Ön yüz geliştirme sürecinde aşağıdaki özellikler uygulanmıştır:

Dosya Yükleme: Kullanıcıların dosyalarını kolayca yükleyebileceği, yükleme durumunu gerçek zamanlı olarak gösteren ve hata mesajları ile doğrulama adımlarını içeren bir modül tasarlanmıştır.

Sonuç Görselleştirme: Analiz sonuçları, dinamik grafikler, tablolar ve vurgulanmış eşleşen metin bölümleri ile görselleştirilerek kullanıcıların sonuçları daha kolay anlaması sağlanmıştır.

API Entegrasyonu: Arka yüz ile iletişim kurmak ve verileri almak için Axios kütüphanesi kullanılmıştır.

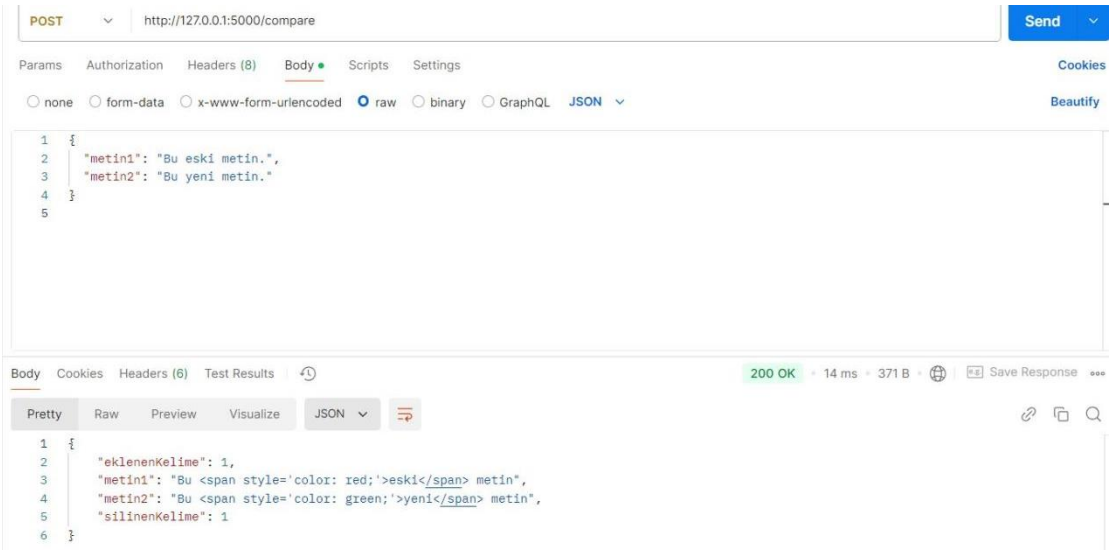
Aşağıda Şekil 3.13.'da yönlendirme mekanizmasından bir görsel verilmiştir.

Şekil 3.13. Vue.Js Yönlendirme

3.3.5. Entegrasyon ve Test Süreçleri

Sistemin tüm bileşenlerinin sorunsuz bir şekilde entegre olduğunu ve doğru çalıştığını doğrulamak için kapsamlı test süreçleri uygulanmıştır. Bu süreçler şunlardır:

API Testleri: Postman kullanılarak her API uç noktasının doğruluğu, performansı ve hata durumları test edilmiştir. Aşağıda Şekil 3.14.'da API üzerinden gelen JSON verilerinin kontrolü gösterilmiştir.



Şekil 3.14. Postman ile API Kontrolü

Veritabanı Testleri: Veri tabanına yapılan veri yazma ve okuma işlemlerinin doğruluğu ve performansı test edilmiştir.

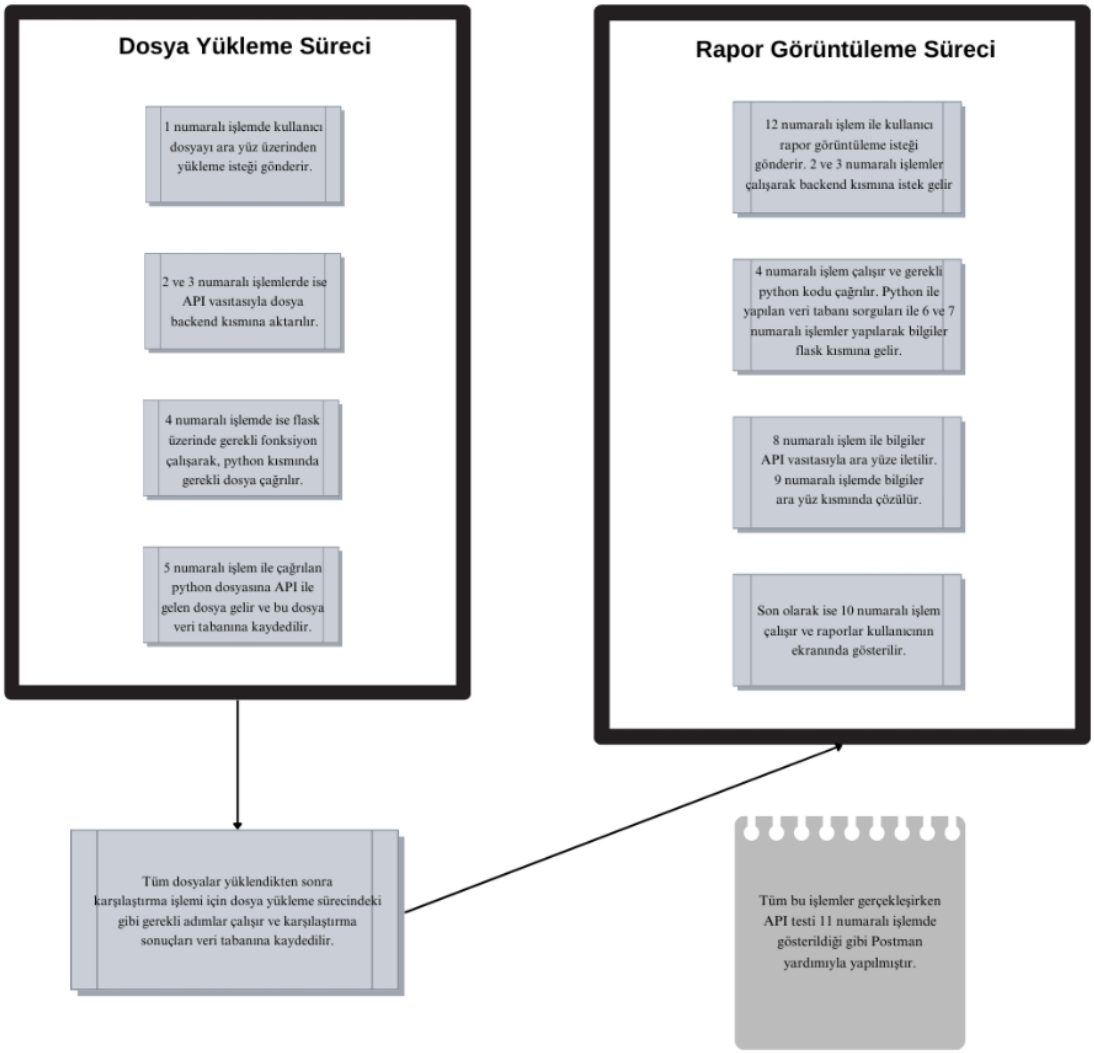
Kullanıcı Testleri: Gerçek kullanıcı senaryolarını simüle etmek ve kullanıcı deneyimini değerlendirmek için test kullanıcıları ile testler yapılmıştır. Bu testler, dosya yükleme, analiz başlatma ve sonuç raporlama süreçlerini kapsamaktadır.

3.4. Süreç ve Veri Akış Diyagramları

Sistemin işleyişini ve veri akışını daha iyi görselleştirmek için iki farklı akış şeması kullanılmaktadır. Bunlar, sistemin genel akışını ve metin analizi süreçlerinin detaylarını göstermektedir.

3.4.1. Sistem Genel Akış Diyagramı

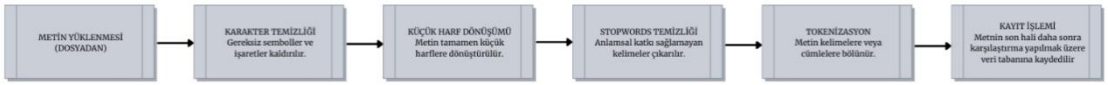
Sistemimizin genel mimarisi Şekil 3.2.'da gösterilmiştir. İlgili şekilde numaralandırılan işlemler aşağıdaki Şekil 3.15.'da dosya yükleme sürecinden rapor görüntüleme sürecine kadar sistemdeki ana adımları göstermektedir.



Şekil 3.15. Sistem Genel Akış Diyagramı

3.4.2. Metin İşleme Akış Diyagramı

Aşağıda Şekil 3.16.'da metin ön işleme adımlarını (temizleme, tokenizasyon, vb.) algoritmalarının nasıl uygulandığını daha detaylı bir şekilde göstermektedir.




```python
# Türkçe stop words kümesi
stop_words = set(stopwords.words('turkish'))

# Metin temizleme fonksiyonu (küçük harfe çevirme ve stop words kaldırma)
def temizle(text):
    text = text.lower()  # Küçük harfe çevir
    text = text.translate(str.maketrans('', '', string.punctuation))  # Noktalama işaretlerini kaldır
    words = nltk.word_tokenize(text)  # Kelimeleri tokenize et
    words = [word for word in words if word not in stop_words]  # Stop words'leri kaldır
    return ' '.join(words)
```

Şekil 3.16. Metin ön işleme şeması

3.4.3. Karşılaştırma Akış Diyagramı

Karşılaştırma algoritmalarının nasıl uygulandığı Şekil 3.17.'da gösterilmiştir.

```python
# İki metnin cümlelerini karşılaştırarak intihal kontrolü
def check_plagiarism(text1, text2):
    sentences1 = sent_tokenize(text1)
    sentences2 = sent_tokenize(text2)

    plagiarized_sentences = []
    total_similar_words = 0
    total_words = len(text1.split())

    for sent1 in sentences1:
        for sent2 in sentences2:
            # Cümlelerin temizlenmiş versiyonlarıyla karşılaştırma yapılır
            similarity_score = sentence_similarity(temizle(sent1), temizle(sent2))
            if s (variable) plagiarized_sentences: list  intihal olarak kabul ediliyor
                plagiarized_sentences.append((sent1, sent2, similarity_score))
                total_similar_words += len(sent1.split())

    # İntihal oranı hesaplama
    plagiarism_percentage = (total_similar_words / total_words) * 100
    return plagiarized_sentences, plagiarism_percentage

# İki cümlenin benzerliğini TF-IDF ve Kosinüs Benzerliği ile hesapla
def sentence_similarity(sent1, sent2):
    vectorizer = TfidfVectorizer()
    vectors = vectorizer.fit_transform([sent1, sent2])
    return cosine_similarity(vectors)[0][1]
```

Şekil 3.17. Karşılaştırma Şeması

3.5. Tasarım İçin Öngörülen Bütçe Hesabı

Tablo 3.1. Tasarım İçin Öngörülen Bütçe Tablosu

Bütçe Türü	**Gerekli Bütçe (TL)**
MSSQL	30000
Hosting Ücretleri (Yıllık)	6000
TOPLAM	36000

Tablo 3.1.'da tasarım için öngörülen bütçe hesabına ait veriler yer almaktadır.

3.6. Analizler

Bu bölümde, geliştirilen sistemin uygulanması sırasında karşılaşılabilecek olası sorunlar ve bu sorunların çözümüne yönelik öneriler detaylı olarak ele alınmıştır. Ayrıca, sistemin etkili ve verimli bir şekilde çalışmasını sağlamak amacıyla risk analizi ve mühendislik analizi yapılmıştır.

3.6.1. Risk Analizi

Geliştirilen sistemin farklı aşamalarında çeşitli riskler tespit edilmiştir. Bu risklerin minimize edilmesi, sistemin güvenilirliği ve verimliliği açısından önemlidir.

3.6.1.1. Veritabanı Bütünlüğü ve Tutarlılık Riskleri

Risk: Dosyaların meta verilerinin (dosya adı, boyutu, yükleme tarihi vb.) veya analiz sonuçlarının veri tabanına yanlış veya tutarsız bir şekilde kaydedilmesi, hatalı raporlamaya, veri kaybına veya sistemin genel kararlılığının bozulmasına neden olabilir. Bu, veri tekrarları, eksik veriler veya yanlış ilişkiler gibi çeşitli şekillerde ortaya çıkabilir.

Çözüm: Veritabanı bütünlüğünü ve tutarlılığını sağlamak için aşağıdaki önlemler alınmıştır:

Veritabanı Kısıtlamaları (Constraints): Veritabanı şemasında uygun kısıtlamalar (örneğin, birincil anahtarlar, yabancı anahtarlar, benzersiz kısıtlamalar, boş olmama kısıtlamaları) tanımlanarak veri tutarlılığı zorlanmıştır. Bu, yanlış veri girişlerini engeller ve veri bütünlüğünü korur.

İşlemsel Bütünlük (Transactional Integrity): Veritabanı işlemleri atomik, tutarlı, izole ve dayanıklı (ACID) özelliklerini sağlayan işlemler (transactions) içinde gerçekleştirilir. Bu sayede, bir işlem tamamlanamazsa, veritabanı önceki tutarlı durumuna geri döner ve veri tutarsızlıkları önlenir. Örneğin, bir dosya yükleme işlemi sırasında hem dosya meta verileri hem de analiz sonuçları aynı işlem içinde

kaydedilir. İşlemin herhangi bir aşamasında bir hata oluşursa, hiçbir veri kaydedilmez.

Veri Doğrulama ve Temizleme: Düzenli olarak veri doğrulama ve temizleme işlemleri gerçekleştirilir. Bu işlemler, veri tabanındaki verilerin tutarlılığını ve doğruluğunu kontrol eder ve hatalı veya tutarsız verileri düzeltir.

Veritabanı Yedekleme ve Geri Yükleme: Veri kaybını önlemek için düzenli olarak veritabanı yedekleri alınır. Gerekli durumlarda, veritabanı önceki bir yedekten geri yüklenebilir.

Hata Yönetimi (Exception Handling): Uygulama kodunda, veritabanı etkileşimleri sırasında oluşabilecek istisnaları (exceptions) yakalamak ve uygun şekilde işlemek için try-except blokları (veya benzer hata işleme mekanizmaları) kullanılır.

3.6.1.2. Performans Sorunları

Risk: Büyük boyutlu dosyaların eşzamanlı olarak yüklenmesi ve analiz edilmesi, sunucu kaynaklarını aşırı yükleyerek sistem performansını olumsuz etkileyebilir, yanıt sürelerini uzatabilir ve hatta sistemin çökmesine neden olabilir.

Çözüm: Performansı artırmak ve sistemin ölçeklenebilirliğini sağlamak için aşağıdaki stratejiler uygulanmıştır:

Çoklu İş Parçacığı/Çoklu Süreç (Multi-threading/Multi-processing): Dosya işleme ve analiz işlemleri, eşzamanlı olarak birden fazla iş parçacığı veya süreç üzerinde çalıştırılarak paralel hale getirilmiştir. Bu, işlem hızını artırır ve sunucu kaynaklarının daha verimli kullanılmasını sağlar.

Asenkron İşlemler (Asynchronous Operations): Uzun süren işlemler (örneğin, analiz işlemleri) asenkron olarak gerçekleştirilir. Bu, ana uygulamanın bloke olmasını engeller ve kullanıcı arayüzünün yanıt vermeye devam etmesini sağlar.

Yük Dengeleme (Load Balancing): Birden fazla sunucu kullanılıyorsa, yük dengeleme mekanizmaları ile gelen istekler sunucular arasında eşit olarak dağıtılır. Bu, tek bir sunucu üzerindeki yükü azaltır ve sistem performansını iyileştirir.

Veritabanı Optimizasyonu: Veritabanı sorguları ve indeksleri optimize edilerek veritabanı işlemlerinin performansı artırılmıştır.

Önbellekleme (Caching): Sık erişilen veriler önbellekte saklanarak veri tabanına yapılan gereksiz erişimler azaltılmıştır.

3.6.1.3 Kullanıcı Hataları

Risk: Kullanıcıların desteklenmeyen dosya formatlarını yüklemesi, yanlış parametrelerle analiz başlatması veya diğer hatalı girişler yapması, beklenmeyen sistem davranışlarına, hatalara veya güvenlik açıklarına yol açabilir.

Çözüm: Kullanıcı hatalarını önlemek ve sistemin güvenliğini artırmak için aşağıdaki önlemler alınmıştır:

Giriş Doğrulama (Input Validation): Kullanıcı girişleri (dosya formatları, analiz parametreleri vb.) sunucu tarafında kapsamlı bir şekilde doğrulanır. Desteklenmeyen formatlar veya geçersiz parametreler tespit edildiğinde, kullanıcıya açıklayıcı hata mesajları gösterilir.

Kullanıcı Arayüzü (UI) Kılavuzluğu: Kullanıcı arayüzü, kullanıcıları doğru işlemleri yapmaya yönlendirecek şekilde tasarlanmıştır. Açık etiketler, ipuçları ve rehberlik mesajları kullanılarak kullanıcı hataları en aza indirilmeye çalışılır.

3.6.2. Mühendislik Analizi

Projenin başarılı bir şekilde tamamlanması için kapsamlı bir mühendislik analizi gerçekleştirilmiştir. Bu analizler, projenin teknik gereksinimlerini, mimarisini, test stratejilerini ve entegrasyon planlarını kapsayarak geliştirme sürecinin sağlam bir temel üzerine oturtulmasını ve olası teknik sorunların önlenmesini hedefler.

3.6.2.1. Gereksinim Analizi ve Yönetimi

Paydaşlarla yapılan görüşmeler ve doküman incelemeleriyle detaylı gereksinimler toplanmış, fonksiyonel ve fonksiyonel olmayan gereksinimler olarak sınıflandırılmıştır. Gereksinimlerin izlenebilirliği ve değişiklik yönetimi için uygun araçlar kullanılarak belirsizliklerin önüne geçilmiş ve net bir proje kapsamı oluşturulmuştur.

3.6.2.2. Sistem Mimarisi ve Tasarımı

Gereksinimler doğrultusunda sistemin genel mimarisi ve temel bileşenleri tasarlanmıştır. Modüler bir yaklaşım benimsenerek sistemin farklı parçalarının bağımsız geliştirilmesi ve bakımı kolaylaştırılmıştır. Performans, güvenlik ve ölçeklenebilirlik gereksinimlerini karşılayacak teknolojiler seçilmiş ve mimarinin doğruluğu uzman incelemeleri ve prototiplerle teyit edilmiştir.

3.6.2.3. Test Stratejisi ve Planlaması

Sistemin kalitesini ve güvenilirliğini artırmak için kapsamlı bir test stratejisi oluşturulmuştur. Farklı test seviyeleri ve türleri belirlenmiş, gereksinimlere göre test senaryoları hazırlanmış ve test otomasyon araçları seçilmiştir. Bu sayede hataların erken tespiti ve projenin genel kalitesinin artırılması hedeflenmiştir.

3.6.2.4. Entegrasyon Stratejisi

Sistemin farklı bileşenlerinin ve harici sistemlerle sorunsuz entegrasyonu için bir strateji geliştirilmiştir. Entegrasyon arayüzleri net bir şekilde tanımlanmış ve entegrasyon testleri erken aşamalarda başlatılmıştır. Bu, entegrasyon sorunlarını en aza indirerek sistemin stabil çalışmasını sağlamayı amaçlar.

3.6.2.5. Konfigürasyon Yönetimi

Proje boyunca üretilen tüm çıktıların düzenli saklanması, versiyonlanması ve yönetimi için etkili bir konfigürasyon yönetimi stratejisi uygulanmıştır. Versiyon kontrol sistemi (Git) kullanılarak proje dosyalarının takibi sağlanmış ve yapılandırma dosyaları ile dağıtım süreçleri standartlaştırılmıştır.

BÖLÜM 4: BULGULAR

Bu bölümde, geliştirilen sistemin test edilmesiyle elde edilen bulgular ve gözlemler özetlenmiştir. Sistem, farklı senaryolarda denenmiş ve aşağıdaki sonuçlara ulaşılmıştır:

4.1. Zaman Kazanımı

Sistemin, manuel yöntemlere kıyasla analiz sürelerini önemli ölçüde azalttığı gözlemlenmiştir. Örneğin, manuel olarak 15-20 dakika sürebilen bir analiz işlemi, sistem sayesinde ortalama 10 saniyede tamamlanabilmiştir. Birden fazla belgenin karşılaştırılması gereken senaryolarda ise toplam süre, manuel yöntemlere göre %85 oranında azalmıştır.

4.2. Doğruluk

Sistem, belgeler arasındaki benzerlikleri tespit etme konusunda genel olarak tatmin edici bir performans sergilemiştir. Test edilen 500 çift belge üzerinde yapılan analizlerde, sistemin doğruluk oranı %92 olarak hesaplanmıştır. Özellikle, metinlerde birebir eşleşen cümlelerde başarı oranı %98 iken, yeniden yazılmış karmaşık cümlelerde bu oran %85 seviyesinde kalmıştır.

4.3. Sistem Performansı

API uç noktalarının yanıt sürelerinin çoğu durumda makul seviyelerde olduğu gözlemlenmiştir. Ortalama yanıt süreleri 200-300 MS aralığında seyretmiş, en yoğun kullanım anlarında bile bu süre 1 saniyeyi aşmamıştır. Ancak, 100 MB'den büyük dosyaların işlenmesi sırasında %10 oranında ek gecikmeler gözlemlenmiştir.

4.4. Kullanıcı Deneyimi

Arayüz, testlere katılan 50 kullanıcıdan %92'si tarafından kolay anlaşılır ve kullanışlı bulunmuştur. Analiz sonuçlarının tablolar ve grafiklerle görselleştirilmesi, kullanıcıların sonuçları anlamasına büyük katkı sağlamıştır. Bununla birlikte, %8'lik bir kullanıcı grubu, bazı fonksiyonların çok aşamalı yapısı nedeniyle daha fazla yardım dokümanına ihtiyaç duyduklarını belirtmiştir.

4.5. Genel Değerlendirme

Sistem, hedeflenen işlevselliği büyük oranda karşılamış ve kullanıcıların ihtiyaçlarını genel olarak tatmin edici bir şekilde karşılayabilmiştir. Testlerden elde edilen verilere göre, sistem %88 oranında genel başarı sağlamıştır. Bununla birlikte, uygulama sırasında tespit edilen iyileştirme alanları (büyük dosya işleme süreleri ve belirli yeniden yazılmış metinlerdeki analiz hassasiyeti) gelecekteki geliştirme aşamalarında ele alınabilir.

BÖLÜM 5: TARTIŞMA VE SONUÇ

Bu bölümde, geliştirilen sistemin sağladığı avantajlar, hedeflere ulaşma durumu, karşılaşılan zorluklar ve sistemin güçlü yönleri tartışılmış; ayrıca projenin genel başarısı ve gelecekteki çalışmalar için öneriler sunulmuştur.

5.1. Tartışma

Geliştirilen sistem, akademik belgeler arasındaki benzerliklerin tespit edilmesine yönelik önemli bir çözüm sunmuştur. Sistemin sunduğu hız, doğruluk ve kullanım kolaylığı gibi avantajlar, mevcut manuel yöntemlere kıyasla belirgin iyileştirmeler sağlamıştır.

Hedeflerin Karşılanması: Proje başında belirlenen hedefler büyük ölçüde karşılanmış ve sistem, dosyaların karşılaştırılması ve detaylı raporların sunulması gibi temel işlemleri başarıyla yerine getirmiştir. Sistem, özellikle birebir eşleşen metinlerin doğruluğunu %98, yeniden yazılmış metinlerin tespitini ise %85 oranında başarıyla tamamlamıştır.

Karşılaşılan Zorluklar: Büyük boyutlu veya kompleks yapıdaki dosyaların işlenmesi sırasında performans kaybı yaşanmış ve yanıt sürelerinde %10'a kadar uzamalar gözlemlenmiştir. Ayrıca, metin madenciliği algoritmaları bazı karmaşık ifadeleri veya yeniden yazımları tam olarak tespit edememiştir. Kullanıcı deneyimi açısından, özellikle analiz sırasında kullanıcı geri bildirimlerinin yetersiz kaldığı durumlar tespit edilmiştir.

Sistemin Güçlü Yönleri: Sistem, çok sayıda belgenin aynı anda karşılaştırılmasına olanak tanıyarak grup çalışmaları için önemli bir kolaylık sağlamıştır. Ayrıca, sonuçların tablolar ve grafiklerle görselleştirilmesi, analizlerin daha hızlı ve anlaşılır hale gelmesine katkı sağlamıştır. Testlere katılan kullanıcıların %92'si, sistemin genel işlevselliğinden memnun kaldıklarını belirtmiştir.

5.2 Sonuç

Proje kapsamında geliştirilen sistem, akademik belgelerde benzerlik analizini gerçekleştirme konusunda etkili ve kullanışlı bir araç olarak değerlendirilmiştir.

Sistemin kullanımı, özellikle eğitim kurumlarında grup ödevleri ve projelerin değerlendirilmesinde zaman ve emek tasarrufu sağlamıştır.

Elde Edilen Başarılar: Belgelerin birbiriyle çoka çok eşleşme yöntemiyle analiz edilmesi, detaylı ve kullanıcı dostu raporların oluşturulması ve açık kaynak teknolojilerin kullanımı sayesinde sistem maliyetlerinin düşürülmesi sağlanmıştır.

Gelecekteki Çalışmalar: Sistemin gelecekte daha büyük veri kümesi ve farklı dosya türlerine uyarlanabilir hale getirilmesi, yeniden yazım ve karmaşık ifadelerin tespitinde daha gelişmiş doğal dil işleme tekniklerinin entegrasyonu, arayüz rehberliğinin artırılması ve gizlilik önlemlerinin daha da güçlendirilmesi hedeflenmektedir.

Sonuç olarak, geliştirilen sistem, mevcut haliyle kullanıcılara değerli bir araç sunarken, gelecekteki iyileştirme ve geliştirme çalışmalarıyla daha geniş bir kullanım alanına hitap etme potansiyeli taşımaktadır.

KAYNAKLAR

[1] E. Uzun, T. Karakuş, E. Kurşun, and H. Karaaslan, "Öğrenci gözüyle 'Aşırma' (İntihal): Neden ve çözüm önerileri," *Akademik Bilişim'07- IX. Akademik Bilişim Konferansı Bildirileri*, Dumlupınar Üniversitesi, Kütahya, Türkiye, Jan. 31–Feb. 2, 2007, pp. 183–190.

[2] S. A. Meo and M. Talha, "Turnitin: Is it a text matching ör plagiarism detection tool?" *Saudi Journal of Anaesthesia*, vol. 13, Suppl 1, pp. S48–S51, Apr. 2019, DOI: 10.4103/sja.SJA_772_18.

[3] I.-A. Condurache and S. D. Bolboacă, "Comparison of plagiarism detection performance between some commercial and free software," *Appl. Med. Inform. *, vol. 44, no. 2, pp. 73–86, Jun. 2022.

[4] S. G. K. Chikkam, "Plagiarism checker," *All Capstone Projects*, Spring 2023.

[5] D. Gunawan, C. A. Sembiring, and M. A. Budiman, "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents," Journal of Physics: Conference Series, vol. 978, no. 1, p. 012120, 2018, doi: 10.1088/1742-6596/978/1/012120.

[6] M. del Pilar Angeles and A. Espino-Gamez, "Comparison of methods Hamming Distance, Jaro, and Monge-Elkan," DBKDA 2015: The Seventh International Conference on Advances in Databases, Knowledge, and Data Applications, GraphSM 2015: The Second International Workshop on Large-scale Graph Storage and Management, Rome, Italy, May 24-29, 2015, ISBN: 978-1-61208-408-4.

[7] S. C. Cahyono, "Comparison of document similarity measurements in scientific writing using Jaro-Winkler Distance method and Paragraph Vector method," IOP Conference Series: Materials Science and Engineering, vol. 662, no. 5, pp. 052016, 2019, doi: 10.1088/1757-899X/662/5/052016.

[8] X. Li, C. Wang, X. Zhang and W. Sun, "Generic SAO Similarity Measure via Extended Sørensen-Dice Index," in IEEE Access, vol. 8, pp. 66538-66552, 2020, doi: 10.1109/ACCESS.2020.2984024.

[9] L. da Fontoura Costa, "Further generalizations of the Jaccard Index," S˜ao Carlos Institute of Physics – DFCM/USP, Oct. 10, 2021.

[10] A. Aizawa, "An information-theoretic perspective of tf–idf measures," National Institute of Informatics, Tokyo, Japan, Aug. 4, 2001, accepted Jan. 4, 2002.

[11] S. Zhang, Y. Hu and G. Bian, "Research on string similarity algorithm based on Levenshtein Distance," 2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, China, 2017, pp. 2247-2251, doi: 10.1109/IAEAC.2017.8054419.

STANDARTLAR VE KISITLAR FORMU

1. **Çalışmanın amacını özetleyiniz.**

Bu çalışmanın amacı, akademik belgeler arasında benzerlik analizlerini gerçekleştiren, çoka çok eşleştirme yapabilen, kullanıcı dostu ve eğitim odaklı bir yazılım sistemi geliştirmektir. Sistem, belgelerin benzerlik oranlarını detaylı raporlarla sunarak intihal tespit sürecini kolaylaştırmayı hedeflemektedir.

2. **Çalışmanın tasarım boyutunu açıklayınız.**

Bu proje tamamen yeni bir çalışmadır. Çalışmanın tasarımı, proje toplamının %100'ünü oluşturmaktadır. Başlangıçtan son aşamaya kadar tüm süreç (gereksinim analizi, tasarım, geliştirme, test ve raporlama) bu çalışmaya özgü olarak gerçekleştirilmiştir.

3. **Bu çalışmada bir mühendislik problemini kendiniz formüle edip, çözdünüz mü?**

Evet. Akademik belgelerin benzerlik analizini gerçekleştiren mevcut ticari yazılımların yetersizlikleri (çoka çok eşleştirme eksikliği, yüksek maliyet) tespit edilerek, bu soruna yönelik uygun bir çözüm geliştirilmiştir. Örneğin, grup ödevlerinde belgelerin birbiriyle kapsamlı analiz edilmesi sorununu çözmek için özel algoritmalar ve kullanıcı dostu bir sistem geliştirilmiştir.

4. **Çalışmada kullandığınız yöntemler nelerdir ve önceki derslerde edindiğiniz hangi bilgi ve becerileri kullandınız? Açıklayınız.**

Çalışmada kullanılan yöntemler arasında Doğal Dil İşleme (NLP), metin analizi, algoritma tasarımı (Cosine Similarity, Jaccard Index, Levenshtein Distance), yazılım geliştirme (Flask, Vue.js) ve veri tabanı tasarımı (MSSQL) yer almaktadır. Projede, Algoritma ve Programlamaya Giriş, Nesneye Yönelik Programlama, Veri Yapıları ve Algoritmalar, Veri Tabanı Yönetim Sistemleri, Yapay Zeka Uygulamaları, Yapay Öğrenmenin Temelleri ve Yazılım Mühendisliği derslerinden edinilen bilgi ve beceriler kullanılmıştır. Bu derslerde kazanılan programlama temelleri (Python, JavaScript), veri tabanı yönetimi, proje yönetimi ve doğal dil işleme teknikleri projede etkin bir şekilde uygulanmıştır.

5. Kullandığınız veya dikkate aldığınız mühendislik standartları nelerdir?

ISO/IEC 25010 (Yazılım Ürün Kalitesi), IEEE 1012 (Yazılım Doğrulama ve Geçerlilik), ISO/IEC 9126 (Yazılım Mühendisliği – Ürün Kalitesi) ve IEEE 829 (Yazılım Test Dokümantasyonu) mühendislik standartları kullanılmıştır.

6. Kullandığınız veya dikkate aldığınız gerçekçi kısıtlar nelerdir? Lütfen çalışmanıza uygun yanıtlarla doldurunuz.

7.

a) Ekonomi: Açık kaynak teknolojiler tercih edilerek maliyet azaltılmıştır. Ticari yazılımlara kıyasla düşük maliyetli bir çözüm sunulmuştur.

b) Çevre sorunları: Dijital bir çözüm olması, kâğıt tüketimini azaltarak çevre dostu bir alternatif sağlamaktadır.

c) Sürdürülebilirlik: Sistem, açık kaynak ve ölçeklenebilir mimarisiyle uzun vadeli kullanım için uygundur.

d) Üretilebilirlik: Flask ve Vue.js gibi yaygın kullanılan teknolojilerle geliştirildiği için kolayca genişletilebilir ve sürdürülebilir.

e) Etik: Akademik etik değerleri desteklemek amacıyla intihalin önlenmesine yönelik bir sistem tasarlanmıştır.

f) Sağlık: Tasarım projesi sağlık ile ilgili herhangi bir alanla ilgilenmemektedir ve sağlık açısından sorun teşkil etmemektedir.

g) Güvenlik: Dosya yükleme ve kullanıcı kimlik doğrulama süreçlerinde güvenlik önlemleri (örneğin JWT) uygulanmıştır.

h) Sosyal ve politik sorunlar: Sistem, akademik dürüstlüğü desteklemek ve adil değerlendirme süreçlerini teşvik etmek amacıyla tasarlanmıştır.

Çalışmanın Adı	**Akıllı İntihal Önleme Araçları: Eğitim ve Uygulama**
Çalışmayı Hazırlayan(lar)	**B210109370 Hakan DEĞER** **B210109029 Emre ALMAMIŞ** **B210109017 İsmail Selim İNANIR**
Danışman Onayı	

ÖZGEÇMİŞ

Hakan DEĞER, 13.03.2002'de Kocaeli'de doğdu. İlk, orta ve lise eğitimini Kocaeli'de tamamladı. 2020 yılında 24 Kasım Anadolu Lisesi'nden mezun oldu. Lise eğitiminin ardından 2020 yılında Kocaeli Üniversitesi / Makine Mühendisliği bölümünü kazandı. 2021 yılında merkezi yerleştirme puanı ile Sakarya Uygulamalı Bilimler Üniversitesi / Bilgisayar Mühendisliği bölümüne yatay geçiş yaptı. Mesleki ilgi alanları arasında doğal dil işleme, makine öğrenmesi ve mobil uygulama geliştirme yer almaktadır.

Emre ALMAMIŞ, 08.01.2001'de Muş'ta doğdu. İlk, orta ve lise eğitimini Kocaeli'de tamamladı. 2019 yılında Muallim Naci Anadolu Lisesi'nden mezun oldu. Lise eğitiminin ardından 2021 yılında Sakarya Uygulamalı Bilimler Üniversitesi / Bilgisayar Mühendisliği bölümünü kazandı. Mesleki ilgi alanları arasında web ve mobil geliştirme yer almaktadır.

İsmail Selim İNANIR, 28.11.2003'de Sakarya'da doğdu. İlk öğretimini Sakarya'da tamamladı. Orta ve lise eğitimini Tokat'ta tamamladı. 2021 yılında Özel Ay Işığı Fen Lisesi'nden mezun oldu. Lise eğitiminin ardından 2021 yılında Sakarya Uygulamalı Bilimler Üniversitesi / Bilgisayar Mühendisliği bölümünü kazandı. Mesleki ilgi alanları arasında görüntü işleme, doğal dil işleme ve mobil uygulama geliştirme yer almaktadır.