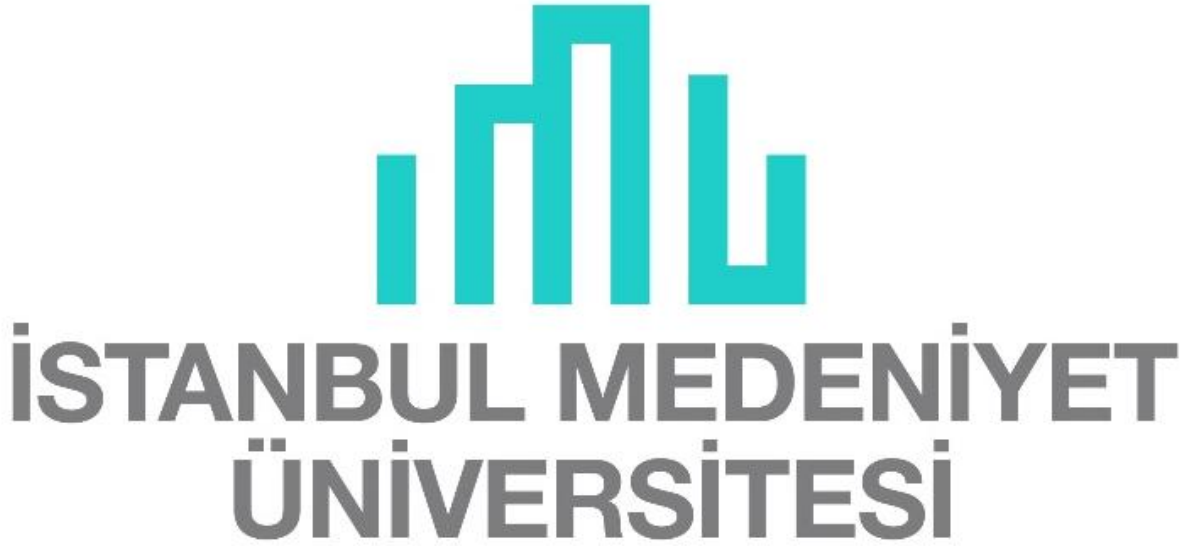


Mühendislik ve Doğa Bilimleri Fakültesi

Bilgisayar Mühendisliği Bölümü

BİL403-YAZILIM MÜHENDİSLİĞİ

Auto Linkedin

Hazırlayanlar;

Alper AKPINAR - 22120205067

Buğrahan ATA - 22120205039

Seferhan KAYA - 22120205048

Öğretim Görevlisi: Dr. Öğr. Üyesi Ertürk ERDAĞI

GitHub: https://github.com/KAYA-Seferhan/Auto-Linkedin

İÇİNDEKİLER

ŞEKİLLER TABLOSU

1. ÖZET

Bu proje çalışmasının amacı, bireylerin sahip oldukları yetkinlikler ve deneyimler doğrultusunda kendilerine en uygun iş ilanlarını daha hızlı, doğru ve etkili bir şekilde bulabilmelerini sağlayan bir yazılım sisteminin geliştirilmesidir. Günümüzde iş arama süreçleri, özellikle çevrimiçi platformlarda çok sayıda ilan arasında manuel filtreleme gerektirmekte ve bu durum kullanıcılar için zaman kaybına ve uygun fırsatların gözden kaçmasına neden olmaktadır. Bu projede ele alınan temel problem, iş arayan bireylerin kişisel profilleri ile iş ilanları arasındaki uyumun otomatik ve nesnel bir biçimde değerlendirilmesidir.

Geliştirilen yazılım, LinkedIn üzerinden elde edilen iş ilanı verilerini temel alarak çalışmaktadır. Kullanıcıdan alınan yetkinlik ve deneyim bilgileri metin tabanlı olarak analiz edilmekte ve bu bilgiler, iş ilanlarının başlık, açıklama ve varsa yetenek gereksinimleri ile karşılaştırılmaktadır. Sistem, metin benzerliğine dayalı bir yaklaşım kullanarak kullanıcı profiline en uygun ilanları sıralı bir şekilde önermektedir. Böylece kullanıcılar, kendi niteliklerine daha yakın pozisyonlara kısa sürede erişebilmektedir.

Proje kapsamında Python programlama dili kullanılarak veri işleme, analiz ve öneri mekanizması geliştirilmiş; ayrıca kullanıcı deneyimini artırmak amacıyla grafiksel bir kullanıcı arayüzü tasarlanmıştır. Geliştirilen arayüz sayesinde kullanıcılar kolayca veri dosyası seçebilmekte, profil bilgilerini girebilmekte ve önerilen iş ilanlarını liste, detay ve bağlantı görünümüyle inceleyebilmektedir.

Elde edilen sonuçlar, geliştirilen sistemin iş ilanları ile kullanıcı profilleri arasında anlamlı eşleştirmeler yapabildiğini ve iş arama sürecini daha verimli hale getirdiğini göstermektedir. Bu yönüyle proje, kariyer yönetimi ve dijital işe alım süreçlerine katkı sağlayabilecek pratik ve geliştirilebilir bir yazılım çözümü sunmaktadır.

2. GİRİŞ

Dijitalleşmenin hız kazanmasıyla birlikte iş bulma ve kariyer yönetimi süreçleri büyük ölçüde çevrimiçi platformlara taşınmıştır. Özellikle LinkedIn gibi profesyonel ağlar, milyonlarca iş ilanını kullanıcılarla buluşturmakta ve bu platformlar hem işverenler hem de iş arayanlar için temel bir kaynak haline gelmektedir. Ancak ilan sayısının giderek artması, kullanıcıların kendi yetkinlik ve deneyimlerine en uygun pozisyonları bulmasını zorlaştırmakta; manuel arama ve filtreleme süreçleri zaman kaybına ve bilgi karmaşasına yol açmaktadır. Bu durum, iş arama sürecinde verimlilik ve doğruluk problemlerini beraberinde getirmektedir.

Gerçek hayatta birçok iş arayan birey, ilan başlıkları veya genel filtreler üzerinden arama yapmakta, ancak ilan açıklamaları ile kendi profil özellikleri arasındaki detaylı uyumu değerlendirmekte zorlanmaktadır. Mevcut sistemlerde sunulan anahtar kelime tabanlı aramalar ve basit filtreleme mekanizmaları, çoğu zaman kullanıcının gerçek yetkinliklerini ve deneyim derinliğini yeterince yansıtamamaktadır. Bu eksiklik, özellikle yeni mezunlar veya kariyer yön değiştiren bireyler için uygun fırsatların gözden kaçmasına neden olmaktadır.

Yazılım mühendisliği alanında bu probleme yönelik olarak öneri sistemleri, bilgi erişim yöntemleri ve metin madenciliği tabanlı yaklaşımlar yaygın biçimde kullanılmaktadır. İçerik tabanlı öneri sistemleri, kullanıcı profilleri ile nesnelerin özelliklerini karşılaştırarak kişiselleştirilmiş sonuçlar üretmeyi amaçlamaktadır [1]. Benzer şekilde, doğal dil işleme ve metin benzerliği yöntemleri, iş ilanı eşleştirme ve aday-pozisyon uyumu değerlendirme çalışmalarında sıklıkla tercih edilmektedir [2]. Ancak birçok mevcut uygulama, bu yöntemleri kullanıcıya şeffaf ve etkileşimli bir arayüzle sunmamakta ya da özelleştirilebilirlik açısından sınırlı kalmaktadır.

Bu proje, mevcut yaklaşımlardan farklı olarak, kullanıcıların kendi yetkinlik ve deneyimlerini serbest metin biçiminde ifade edebildiği ve bu bilgilerin gerçek iş ilanı verileriyle otomatik olarak karşılaştırıldığı bir yazılım sistemi geliştirmeyi hedeflemektedir. Ayrıca geliştirilen sistem, yalnızca algoritmik eşleştirme sunmakla kalmayıp, kullanıcı dostu bir grafiksel arayüz aracılığıyla sonuçların kolayca incelenebilmesini sağlamaktadır. Bu yönüyle proje, teknik doğruluk ile kullanıcı deneyimini bir arada ele alan bütüncül bir çözüm sunmaktadır.

Raporun devamında, üçüncü bölümde gereksinim analizi ve problem tanımı detaylandırılmakta, dördüncü bölümde süreç modeli ve planlama açıklanmaktadır. Beşinci bölümde yazılım tasarımı ve mimarisi ele alınırken, altıncı bölümde gerçeklenen yazılım ve detaylar sunulmaktadır. Yedinci bölümde test süreçlerine yer verilirken, sekizinci bölümde yazılım kalitesi ve güvenliği incelenmektedir. Son bölümde ise sonuçlar ele alınmaktadır.

3. GEREKSİNİM ANALİZİ

3.1. Problem Tanımı

Bu proje kapsamında ele alınan temel problem, iş arayan bireylerin sahip oldukları yetkinlikler ve deneyimler doğrultusunda kendilerine en uygun iş ilanlarını etkili bir şekilde bulmakta yaşadıkları zorluklardır. Günümüzde LinkedIn gibi çevrimiçi iş platformları çok sayıda ilan sunmakta, ancak bu ilanların büyük bir kısmı kullanıcılar için anlamlı ve uygun değildir. Mevcut sistemlerde sunulan anahtar kelime aramaları ve basit filtreleme mekanizmaları, kullanıcı profilleri ile iş ilanları arasındaki gerçek uyumu yeterince yansıtamamaktadır.

Bu durum, kullanıcıların uzun süre ilan incelemesine rağmen uygun pozisyonlara ulaşamamasına, zaman kaybına ve motivasyon kaybına neden olmaktadır. Özellikle yeni mezunlar, kariyer yön değiştirenler veya teknik becerileri detaylı olan kullanıcılar için bu problem daha belirgin hale gelmektedir.

Geliştirilen yazılım, bu sorunu çözmek amacıyla kullanıcı yetkinlikleri ile iş ilanlarını metin tabanlı olarak analiz eden ve aralarındaki benzerliği hesaplayarak en uygun ilanları öneren bir sistem sunmaktadır. Böylece kullanıcıların iş arama süreci daha verimli, hızlı ve kişiselleştirilmiş hale getirilmektedir.

3.2. Hedef Kullanıcılar

Geliştirilen sistemin hedef kullanıcı grupları aşağıda listelenmiştir:

- **Yeni mezunlar:** Sınırlı iş deneyimine sahip, yetkinlik odaklı iş arayan bireyler
- **Deneyimli profesyoneller:** Kariyerinde ilerlemek veya pozisyon değiştirmek isteyen kullanıcılar
- **Staj arayan öğrenciler:** Uygun staj ve başlangıç seviyesi pozisyonları arayan kullanıcılar

Bu kullanıcıların sistemden temel beklentileri; kendilerine uygun ilanların hızlıca listelenmesi, ilan detaylarının açık bir şekilde görüntülenmesi ve karmaşık arama süreçleriyle uğraşmadan anlamlı öneriler alabilmektir.

3.3. Fonksiyonel Gereksinimler

Sistemin sağlaması beklenen temel işlevler aşağıda sıralanmıştır:

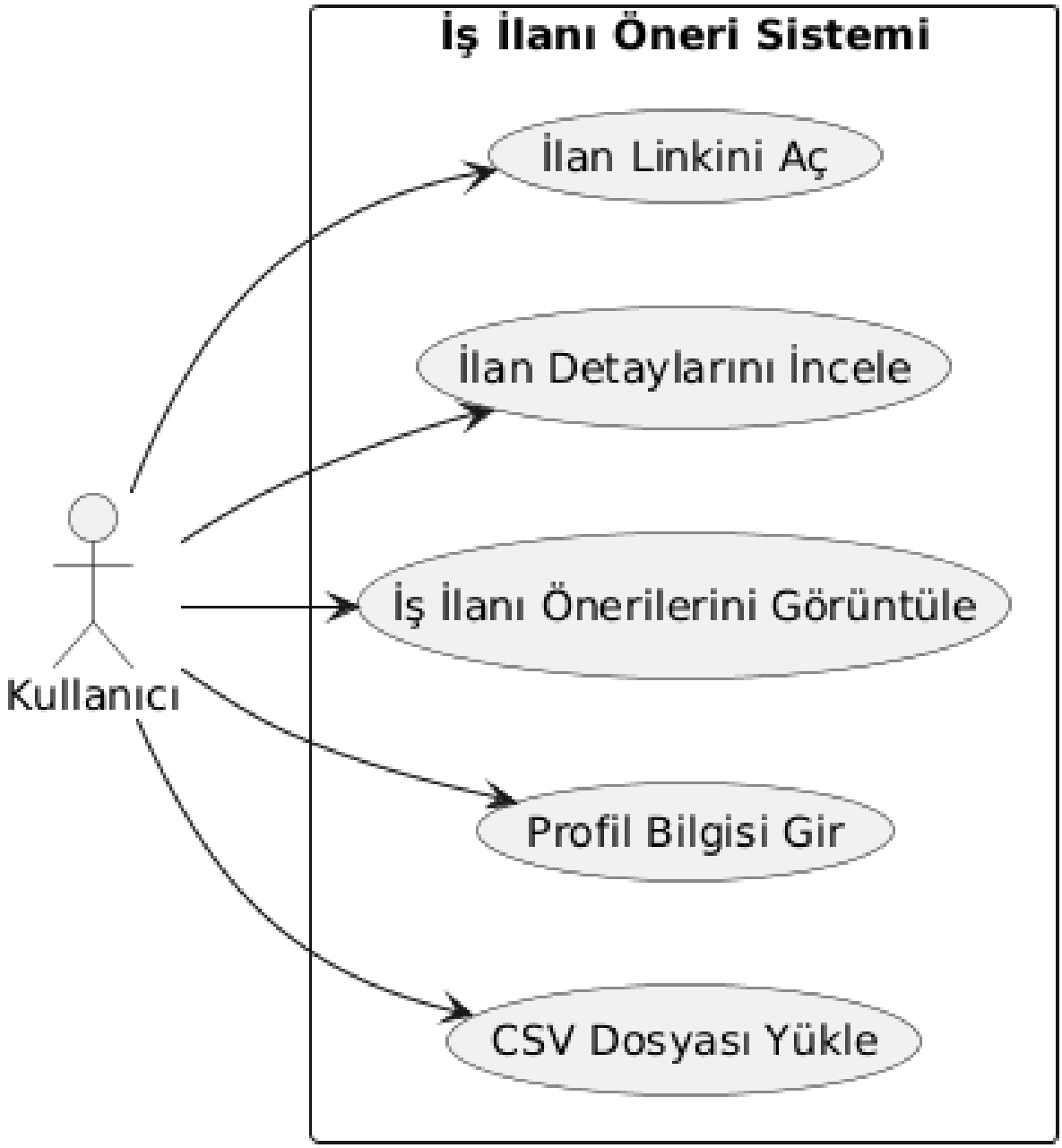
- Kullanıcının LinkedIn iş ilanlarını içeren bir CSV dosyasını sisteme yükleyebilmesi
- Kullanıcının yetkinlik ve deneyimlerini serbest metin olarak girebilmesi
- Kullanıcı profili ile iş ilanları arasında benzerlik analizi yapılması
- En uygun iş ilanlarının benzerlik skoruna göre sıralanarak listelenmesi
- Kullanıcının ilan detaylarını görüntüleyebilmesi
- Seçilen iş ilanının LinkedIn bağlantısının tarayıcıda açılabilmesi

3.4. Fonksiyonel Olmayan Gereksinimler

Sistemin kalite özellikleri kapsamında aşağıdaki gereksinimleri karşılaması beklenmektedir:

- **Performans:** Orta ölçekli iş ilanı veri setleri üzerinde kısa sürede öneri üretebilmelidir
- **Kullanılabilirlik:** Grafiksel kullanıcı arayüzü sade, anlaşılır ve kullanıcı dostu olmalıdır
- **Güvenlik:** Kullanıcıdan alınan veriler yalnızca yerel ortamda işlenmeli ve dış sistemlerle paylaşılmamalıdır
- **Taşınabilirlik:** Sistem farklı işletim sistemlerinde (Windows, Linux) çalışabilir olmalıdır
- **Bakım Kolaylığı:** Kod yapısı modüler ve geliştirilebilir olmalıdır

3.5. Use-Case Diagram



Şekil 1. UML Standartlarına Uygun Use-Case Diagram

Use-Case Diagram'da sistemin tek aktörü Kullanıcı olarak tanımlanmıştır. Kullanıcı, sisteme iş ilanlarını içeren CSV dosyasını yükleyerek süreci başlatmaktadır. Ardından yetkinlik ve deneyim bilgilerini sisteme girmekte ve sistem tarafından üretilen iş ilanı önerilerini görüntüleyebilmektedir. Kullanıcı, önerilen ilanların detaylarını inceleyebilmekte ve dilerse ilgili ilanı harici bir tarayıcı üzerinden açabilmektedir. Diyagram, sistemin kullanıcı odaklı ve etkileşimli yapısını açık bir şekilde ortaya koymaktadır.

4. SÜREÇ MODELİ

Bu projede yazılım geliştirme süreci olarak Çevik Yazılım Geliştirme Yaklaşımı benimsenmiş ve uygulamada Scrum modeli kullanılmıştır. Scrum, kısa süreli geliştirme döngüleri (sprintler) aracılığıyla artımlı ve yinelemeli ürün geliştirmeyi esas alan bir çevik yöntemdir. Bu model; değişen gereksinimlere hızlı uyum sağlanabilmesi, erken geri bildirim alınabilmesi ve geliştirilen ürünün sürekli iyileştirilmesine olanak tanıması gibi özellikleriyle öne çıkmaktadır.

Seçilen Scrum modelinin bu proje için uygun olmasının temel nedeni, projenin sınırlı zaman aralığında geliştirilmesi, ekip yapısının küçük olması ve gereksinimlerin proje süreci içerisinde netleşerek olgunlaşmasıdır. İş ilanı öneri sistemi gibi kullanıcı etkileşimi yüksek olan bir yazılımda, kullanıcı arayüzü ve öneri mantığının geliştirme sürecinde tekrar tekrar değerlendirilmesi ve iyileştirilmesi önemli görülmüştür. Scrum modeli, bu tür iteratif geliştirme ihtiyaçlarını karşılayabilecek esnekliği sağlamaktadır.

4.1. Sprint Yapısı ve Zaman Planı

Proje toplamda üç sprint halinde planlanmış ve her sprint iki haftalık bir süreyi kapsamıştır. Sprintlerin genel amaçları aşağıda özetlenmiştir:

- **Sprint 1 – Analiz ve Temel Altyapı:**

Problem analizi, gereksinimlerin belirlenmesi, veri yapılarının incelenmesi ve sistem mimarisinin oluşturulması. Bu sprintte ayrıca temel veri okuma ve ön işleme fonksiyonları geliştirilmiştir.

- **Sprint 2 – Geliştirme:**

Metin tabanlı öneri mekanizmasının geliştirilmesi, benzerlik hesaplama yöntemlerinin uygulanması ve sistemin çekirdek işlevlerinin tamamlanması. Kullanıcıdan alınan profil bilgilerinin iş ilanlarıyla karşılaştırılması bu sprintte gerçekleştirilmiştir.

- **Sprint 3 – Arayüz ve Test:**

Grafiksel kullanıcı arayüzünün tasarlanması, sistem bileşenlerinin entegrasyonu ve fonksiyonel testlerin gerçekleştirilmesi. Kullanılabilirlik iyileştirmeleri ve hata düzeltmeleri bu aşamada yapılmıştır.

4.2. Süreç Faaliyetleri

Geliştirme süreci boyunca analiz, tasarım, geliştirme ve test faaliyetleri çevik yaklaşım doğrultusunda iç içe yürütülmüştür. Analiz aşamasında kullanıcı ihtiyaçları ve problem kapsamı belirlenmiş; tasarım aşamasında sistem mimarisi ve kullanıcı etkileşimleri planlanmıştır. Geliştirme sürecinde işlevler artımlı olarak implemente edilmiş ve her sprint sonunda çalışan bir ürün ortaya konmuştur. Test faaliyetleri ise her sprint sonunda gerçekleştirilmiş; fonksiyonel testler ve kullanıcı senaryoları üzerinden sistemin doğruluğu ve kararlılığı değerlendirilmiştir.

Bu yapı sayesinde proje süreci kontrollü, esnek ve sürdürülebilir bir şekilde ilerlemiş; geliştirilen yazılımın hem teknik gereksinimleri hem de kullanıcı beklentilerini karşılaması sağlanmıştır.

5. YAZILIM TASARIMI VE MİMARİ

5.1. Mimari Yaklaşım

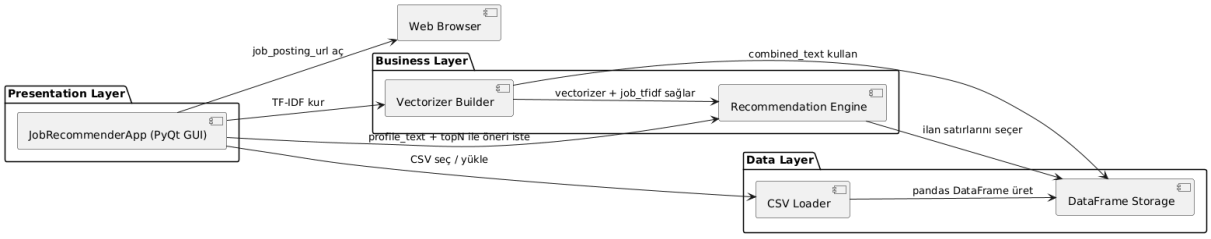
Geliştirilen uygulama, katmanlı bir mimari anlayışıyla ve MVC'ye benzer bir ayrım gözetilerek tasarlanmıştır. Uygulama tek bir masaüstü programı olarak çalışsa da, kod yapısı üç temel sorumluluk etrafında ayrıştırılmıştır:

1. **Veri Katmanı (Data Layer):** CSV dosyasından iş ilanı verisinin okunması ve temizlenmesi
2. **İş Mantığı Katmanı (Business/Recommendation Layer):** Metin temsili oluşturma, TF-IDF çıkarımı ve benzerlik hesaplama ile öneri üretimi
3. **Sunum Katmanı (Presentation/UI Layer):** Kullanıcı arayüzü üzerinden dosya seçimi, profil girişi, önerilerin listelenmesi ve detayların gösterimi

Bu yaklaşım sayesinde; sistemin okunabilirliği artmış, yeni özellik ekleme (filtreleme, farklı öneri algoritmaları vb.) daha yönetilebilir hale gelmiştir.

5.2. Sistem Bileşenleri ve Component Diagram

Sistem; kullanıcı arayüzü bileşenleri, öneri motoru ve veri işleme bileşenleri üzerinden çalışmaktadır. Kullanıcı, CSV dosyasını seçerek veri kümesini sisteme yükler; ardından profil metnini girer ve öneri listesi oluşturulur. Öneri çıktısı, tabloda listelenir ve seçilen ilanın detayları ayrı bir panelde gösterilir.

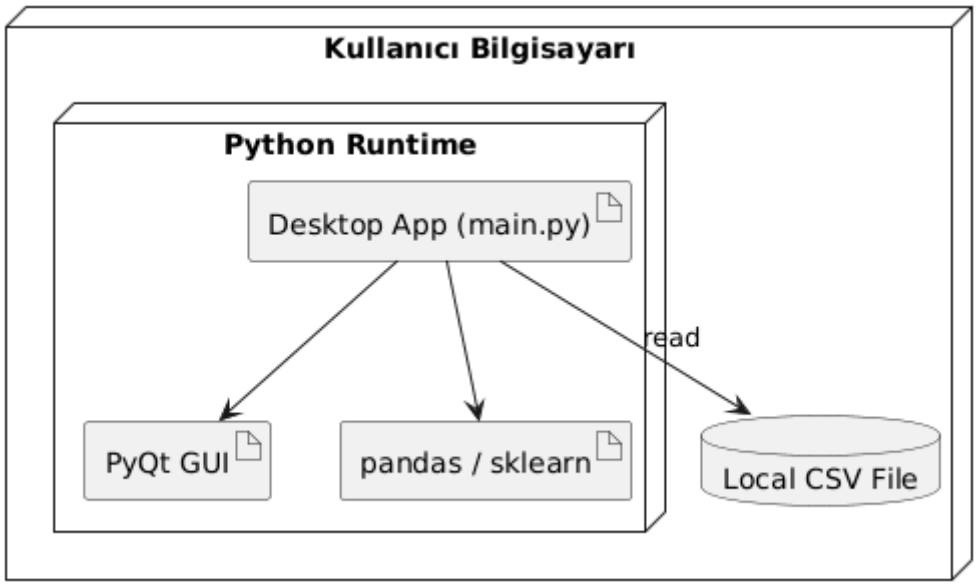


Şekil 2. UML Component Diagram

- `JobRecommenderApp`, kullanıcıyla etkileşimin gerçekleştiği ana bileşendir.
- `CSV Loader`, iş ilanı verisini dosyadan okur ve `DataFrame` olarak hazırlar.
- `Vectorizer Builder`, metinleri TF-IDF vektörlerine dönüştürmek için vektörleştiriciyi oluşturur.
- `Recommendation Engine`, kullanıcı profil metni ile ilan vektörleri arasında benzerlik hesaplar ve en uygun ilanları döndürür.
- Seçilen ilanın bağlantısı Web Browser üzerinden açılır.

5.3. Deployment Yaklaşımı

Uygulama masaüstünde yerel olarak çalıştığından dağıtım modeli basittir. Çalışma zamanı bileşenleri; Python yorumlayıcısı ve bağımlılık kütüphaneleri üzerinde çalışır. Veri kaynağı yerel CSV dosyasıdır. Harici servis entegrasyonu yoktur (bağlantı açma yalnızca tarayıcı çağrısıdır).



Şekil 3. UML Deployment Diagram

5.4. Veri Yapıları ve Veri Modeli

Bu sistemde klasik bir veritabanı (SQL/NoSQL) kullanılmamıştır. Veri modeli, CSV tabanlı bir yapı üzerinden yönetilmektedir:

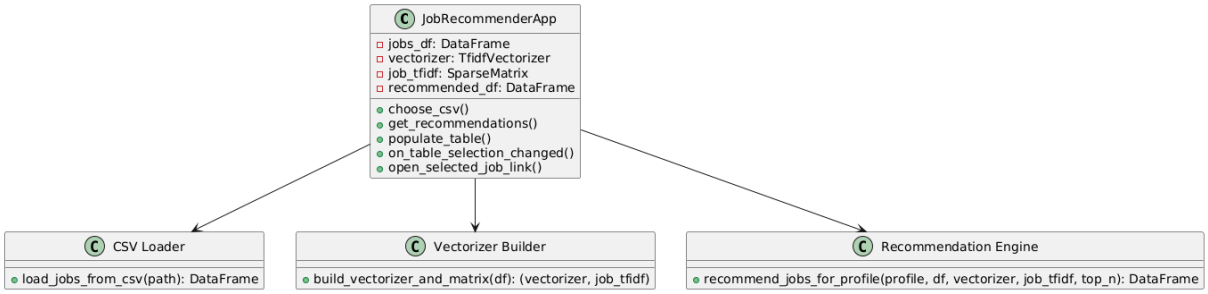
- İş ilanları CSV'den okunarak pandas.DataFrame içerisinde saklanır.
- Modelleme için kullanılan temel alanlar:
    - title, description, skills_desc (varsa) → combined_text oluşturulur.
- Öneri motoru için ara veri yapıları:
    - vectorizer: TfidfVectorizer
    - job_tfidf: ilanların TF-IDF matrisi (sparse matrix)
    - profile_vec: kullanıcının TF-IDF vektörü
    - similarities: cosine similarity sonuçları (1 boyutlu dizi)

Bu tasarım, küçük/orta ölçekli veri setlerinde hızlı prototipleme ve kolay kullanım sağlamaktadır.

yüklemekten sorumludur. JobRecommender sınıfı ise sistemin çekirdek iş mantığını oluşturarak kullanıcı profili ile iş ilanları arasında eşleştirme yapmaktadır. Sınıflar arasındaki ilişkiler, sistemin modüler ve sorumluluk temelli tasarlandığını göstermektedir.

5.5. Temel Sınıflar ve Sınıf İlişkileri

Uygulamanın sınıf temelli ana bileşeni JobRecommenderApp sınıfıdır. Bu sınıf UI olaylarını yönetir ve yardımcı fonksiyonları çağırarak veri akışını kontrol eder. Öneri motoru ve veri yükleme işlemleri fonksiyonlar halinde modülerleştirilmiştir.



Şekil 4. UML Class Diagram

5.6. Kullanıcı Arayüzü Tasarımı ve UX Gerekçeleri

Uygulamanın arayüzü, kullanıcıların minimum adımla sonuç alabileceği şekilde tasarlanmıştır. Arayüz üç ana aşamayı destekler:

- **CSV seçimi:** Kullanıcı ilan verisini dosya seçici ile yükler ve sistemin hazır hale geldiğini durum etiketi üzerinden görür.
- **Profil metni girişi:** Kullanıcı yetkinlik/deneyim bilgisini serbest metin olarak girer.
- **Öneri ve inceleme:** Öneriler tabloda listelenir; bir ilan seçildiğinde sağ panelde açıklama ve bağlantı görüntülenir.

Tasarım kararları (kullanılabilirlik gerekçesi):

- **Adım adım akış:** “1-2-3” şeklinde gruplanmış kutular, kullanıcıyı süreçte yönlendirir.
- **Split görünüm:** Sol tarafta liste, sağ tarafta detay gösterimi; kullanıcı bağlamı kaybetmeden ilanları inceleyebilir.
- **Tıklanabilir bağlantı:** “İlanı tarayıcıda aç” butonu ile aksiyon tek adımda tamamlanır.
- **Koyu tema:** Uzun süre ilan inceleme senaryolarında göz yorgunluğunu azaltmayı hedefler.

6. GERÇEKLEŞTİRİLEN YAZILIM

Bu bölümde geliştirilen yazılımın hayata geçirilme süreci, kullanılan teknolojiler, modüler yapı, kritik kod bileşenleri ve kullanıcı etkileşimleri somut biçimde açıklanmaktadır.

6.1. Kullanılan Teknolojiler ve Tercih Nedenleri

Proje geliştirme sürecinde aşağıdaki teknolojiler kullanılmıştır:

- **Python:**

Projenin ana programlama dili olarak Python tercih edilmiştir. Python; veri işleme, metin analizi ve makine öğrenmesi alanlarında sunduğu zengin kütüphane ekosistemi, hızlı prototipleme imkânı ve okunabilir sözdizimi nedeniyle seçilmiştir.

- **PyQt5:**

Masaüstü kullanıcı arayüzünün geliştirilmesinde PyQt5 framework'ü kullanılmıştır. PyQt5, modern ve etkileşimli arayüzler geliştirmeye olanak sağlaması, olay tabanlı programlama yapısını desteklemesi ve Python ile güçlü entegrasyonu nedeniyle tercih edilmiştir.

- **pandas:**

LinkedIn iş ilanlarının CSV formatındaki verilerinin okunması, temizlenmesi ve yönetilmesi amacıyla kullanılmıştır. Büyük veri setleri üzerinde esnek ve hızlı işlem yapılmasına olanak tanımaktadır.

- **scikit-learn:**

Metinlerin sayısal temsile dönüştürülmesi ve benzerlik hesaplamaları için kullanılmıştır. TfidfVectorizer ve cosine_similarity fonksiyonları, öneri sisteminin temelini oluşturmaktadır.

- **Geliştirme Ortamı:**

Geliştirme süreci Windows işletim sistemi üzerinde, Visual Studio Code / PyCharm benzeri Python IDE'leri kullanılarak gerçekleştirilmiştir.

6.2. Modüler Yapı ve Modüllerin İşlevleri

Yazılım, fonksiyonel sorumluluklara göre modüler bir yapı ile geliştirilmiştir. Tek dosya içerisinde yer alsa da mantıksal olarak aşağıdaki modüllerden oluşmaktadır:

6.2.1. Veri Yükleme Modülü (load_jobs_from_csv)

- CSV dosyasını okur
- Gerekli kolonları kontrol eder
- Eksik verileri temizler
- Metin alanlarını birleştirerek analiz için hazır hale getirir

6.2.2. Metin Vektörleştirme Modülü (build_vectorizer_and_matrix)

- İş ilanı metinlerinden TF-IDF vektörleri üretir
- Modelin yeniden kullanılabilmesi için vektörleştiriciyi saklar

6.2.3. Öneri Motoru (recommend_jobs_for_profile)

- Kullanıcı profil metnini vektörleştirir
- Cosine similarity ile benzerlik skorlarını hesaplar
- En uygun ilanları sıralı biçimde döndürür

6.2.4. Kullanıcı Arayüzü Modülü (JobRecommenderApp)

- Tüm GUI bileşenlerini yönetir
- Kullanıcı etkileşimlerini kontrol eder
- Öneri sonuçlarını tablo ve detay görünümü şeklinde sunar

6.3. Kritik Kod Parçaları ve Açıklamaları

- TF-IDF ve Benzerlik Hesaplama

Aşağıda, sistemin temel öneri mantığını oluşturan temsil edici bir kod parçası yer almaktadır:

```
profile_vec = vectorizer.transform([profile_text])
similarities = cosine_similarity(profile_vec, job_tfidf)[0]
top_indices = similarities.argsort()[::-1][:top_n]
```

Bu kod parçası; kullanıcıdan alınan profil metnini TF-IDF vektörüne dönüştürmekte, tüm iş ilanlarıyla olan benzerlik skorlarını hesaplamakta ve en yüksek skora sahip ilanları seçmektedir. Sistem, önerilerini tamamen bu hesaplama sonucuna göre oluşturmaktadır.

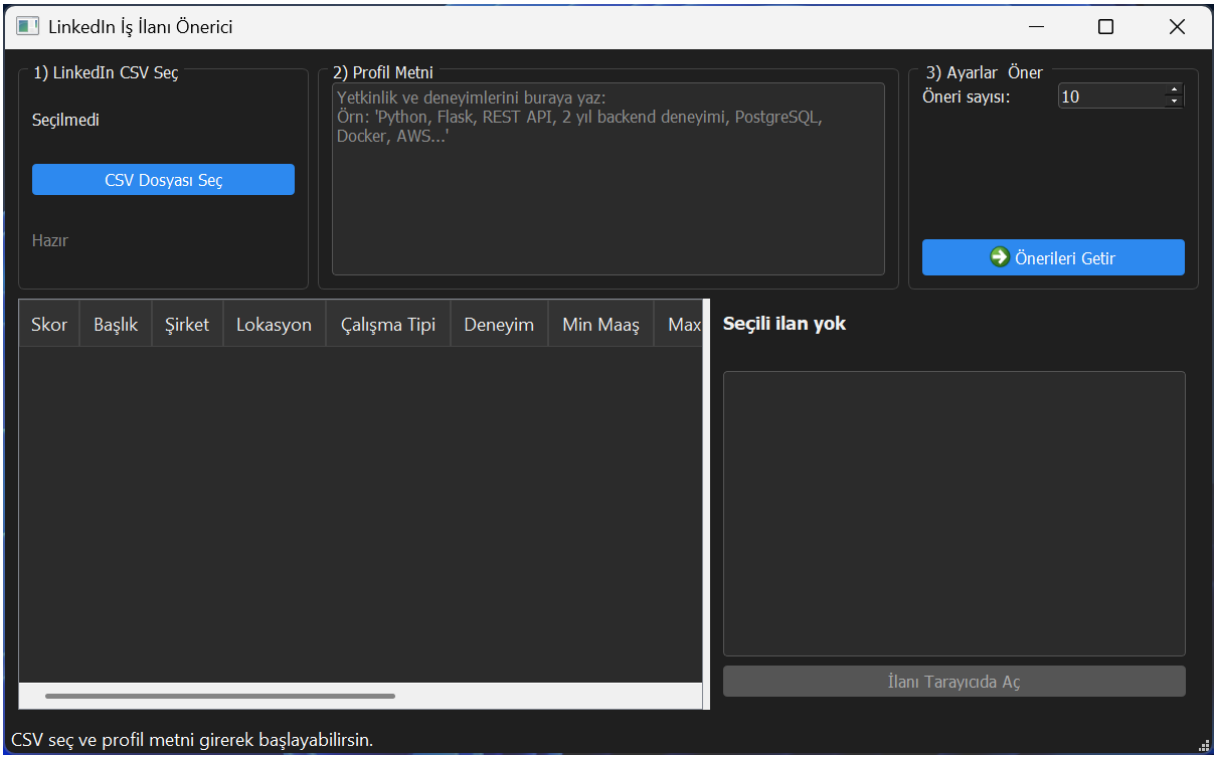
- CSV Yükleme ve Metin Birleştirme

```
df["combined_text"] = df["title"] + " " + df["description"] + " " +
df["skills_desc"]
```

Bu yapı sayesinde ilan başlığı, açıklaması ve varsa yetenek alanları tek bir metin haline getirilerek daha anlamlı ve kapsamlı bir içerik analizi yapılmaktadır.

6.4. Kullanıcı Arayüzü ve Ekran Görüntüleri

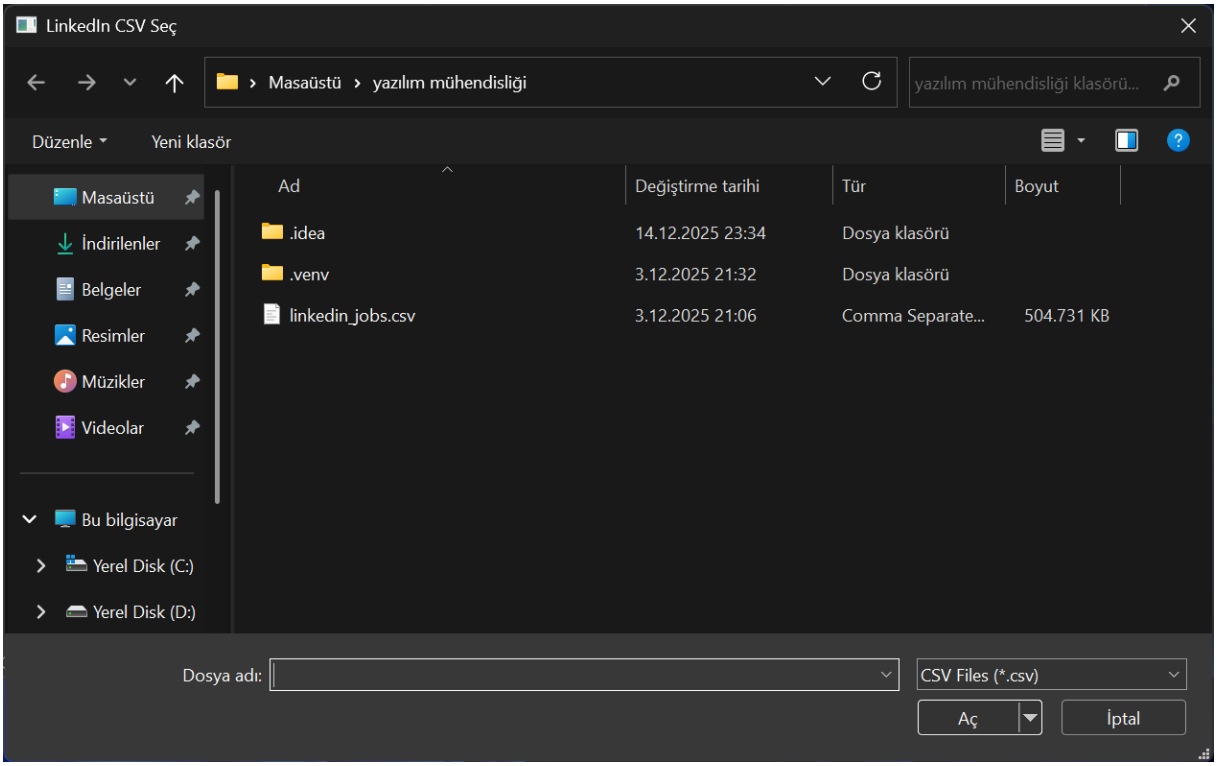
Uygulama çalışır durumda masaüstü ortamında aşağıdaki ekranlardan oluşmaktadır:



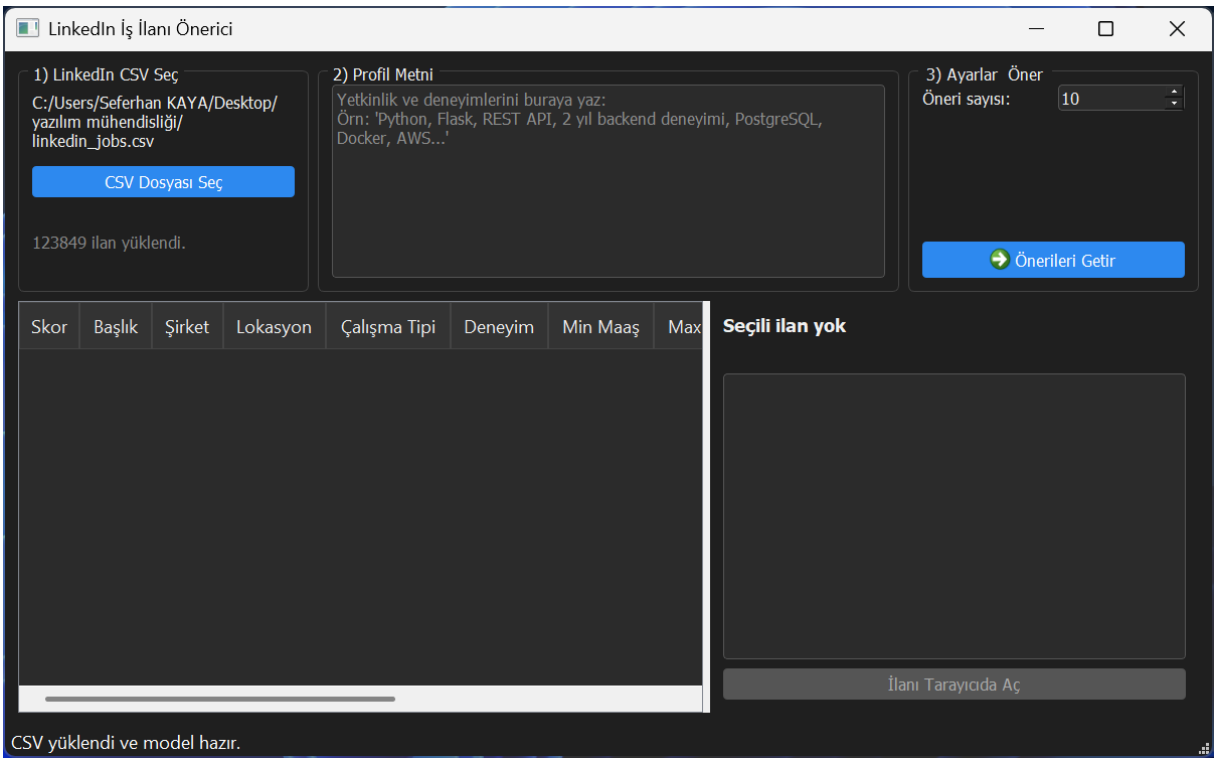
Şekil 5. Ana Sayfa

- **Ekran 1 – CSV Seçim Ekranı:**

Kullanıcı LinkedIn iş ilanı verisini CSV dosyası olarak seçer.



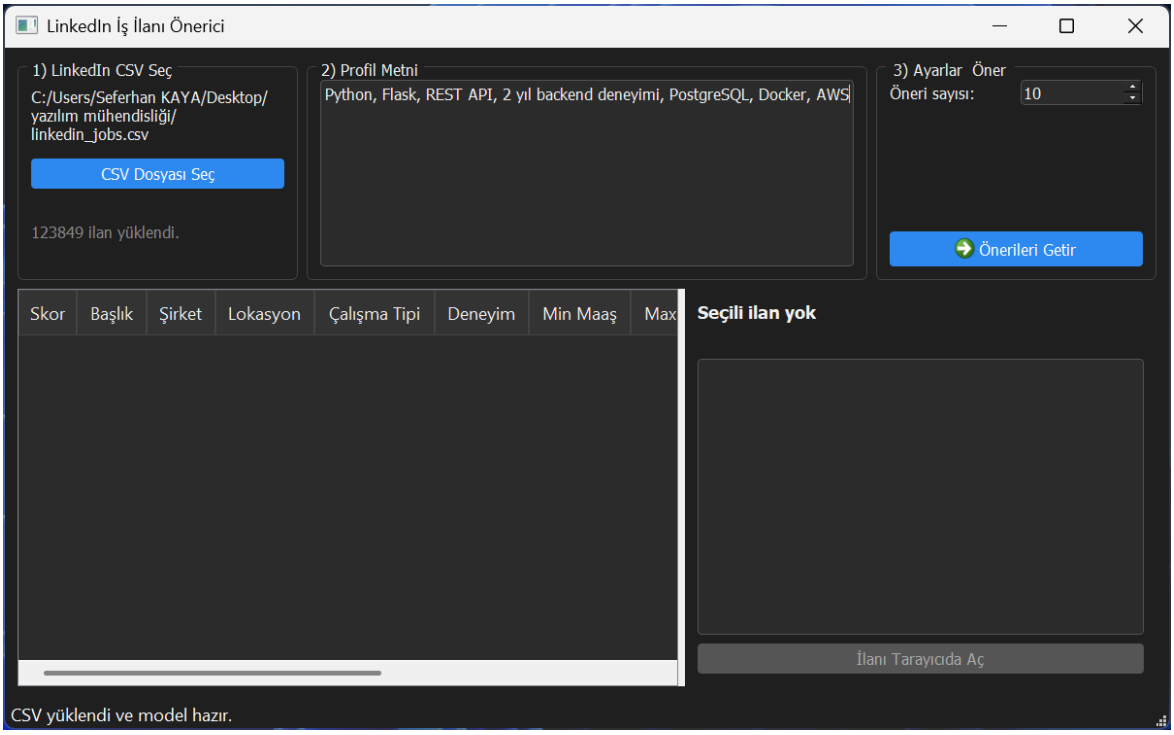
Şekil 6. CSV Seçim Ekranı



Şekil 7. CSV Yüklendi

- **Ekran 2 – Profil Girişi:**

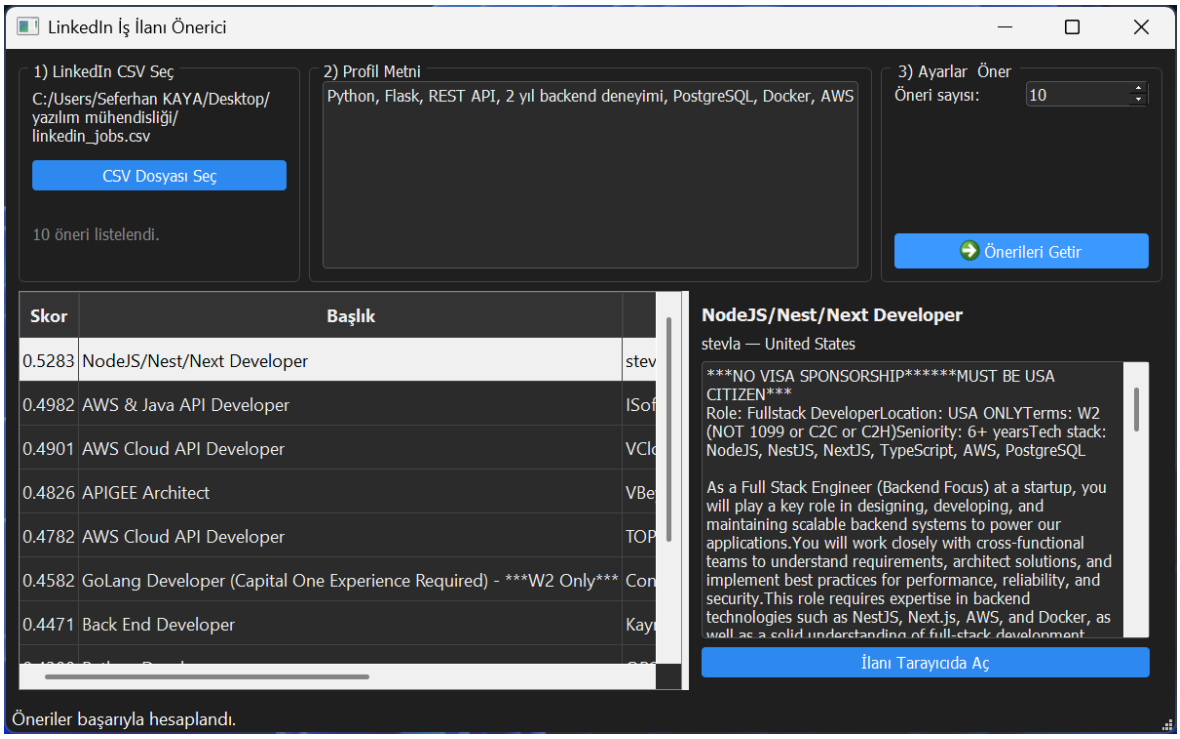
Kullanıcı yetkinlik ve deneyimlerini serbest metin olarak sisteme girer.



Şekil 8. Profil Girişi

- **Ekran 3 – Öneri Listesi ve İlan Detayı:**

Sistem, benzerlik skorlarına göre önerilen ilanları tablo halinde listeler.



Şekil 9. Öneri Listesi ve İlan Detayı

6.5. Kullanıcı Senaryosu

1. Kullanıcı uygulamayı başlatır.
2. LinkedIn'den elde edilen CSV dosyasını seçer.
3. Profil metnini (yetkinlikler, deneyim) girer.
4. "Önerileri Getir" butonuna tıklar.
5. Sistem en uygun ilanları listeler.
6. Kullanıcı bir ilanı seçerek detaylarını inceler ve isterse tarayıcıda açar.

Bu senaryo, gerçek hayattaki iş arama sürecini birebir destekleyecek şekilde tasarlanmıştır.

7. TEST SÜREÇLERİ

7.1. Test Yaklaşımı

Proje kapsamında dinamik test yaklaşımı benimsenmiştir. Testler, yazılım çalışır durumdayken gerçekleştirilmiş ve gerçek kullanıcı senaryoları üzerinden değerlendirilmiştir. Çevik geliştirme süreciyle uyumlu olacak şekilde test faaliyetleri her sprint sonunda uygulanmış, böylece geliştirilen işlevlerin doğruluğu sürekli olarak kontrol edilmiştir.

Test kapsamı; sistemin çekirdek işlevleri olan veri yükleme, öneri üretme, sonuçların görüntülenmesi ve kullanıcı etkileşimi üzerine odaklanacak şekilde belirlenmiştir.

7.2. Uygulanan Test Türleri

Projede aşağıdaki test türleri uygulanmıştır:

- **Birim Testleri (Unit Testing):**

Sistemin temel fonksiyonlarının doğru çalışıp çalışmadığını doğrulamak amacıyla uygulanmıştır.

- **Fonksiyonel Testler:**

Kullanıcıdan alınan girdilere karşılık sistemin beklenen çıktıları üretip üretmediği kontrol edilmiştir.

- **Entegrasyon Testleri:**

Veri işleme, öneri motoru ve kullanıcı arayüzü modüllerinin birlikte uyumlu çalışıp çalışmadığı test edilmiştir.

- **Kullanılabilirlik Testleri:**

Arayüzün anlaşılabilirliği, kullanıcı akışlarının sezgisel olup olmadığı ve etkileşim kolaylığı değerlendirilmiştir.

7.3. Birim Testi Örnekleri

Aşağıda, öneri motorunda kullanılan benzerlik hesaplama fonksiyonuna ait örnek bir birim testi sunulmaktadır:

```
def test_similarity_output_range():
    similarities = cosine_similarity(profile_vec, job_tfidf)[0]
    assert all(0.0 <= score <= 1.0 for score in similarities)
```

Bu testte, kullanıcı profili ile iş ilanları arasında hesaplanan benzerlik skorlarının 0 ile 1 arasında olması beklenmektedir. Testin başarılı olması, cosine similarity fonksiyonunun doğru ve beklenen aralıkta sonuç ürettiğini göstermektedir.

Bir diğer örnek test, CSV dosyasından veri yükleme fonksiyonuna yöneliktir:

```
def test_csv_loader_not_empty():
    df = load_jobs_from_csv("sample_jobs.csv")
    assert len(df) > 0
```

Bu testte, geçerli bir CSV dosyasının sisteme yüklendiğinde boş olmayan bir veri yapısı oluşturması beklenmektedir. Böylece veri yükleme modülünün temel işlevi doğrulanmaktadır.

7.4. Karşılaşılan Hatalar ve Çözüm Süreci

Geliştirme süreci boyunca çeşitli teknik ve kullanım kaynaklı hatalarla karşılaşılmıştır. Bunlardan bazıları aşağıda özetlenmiştir:

- **CSV Alan Eksikliği Hataları:**

Bazı CSV dosyalarında beklenen kolonların eksik olması, sistemin çalışmasını engellemiştir. Bu sorun, veri yükleme aşamasında kolon kontrolü ve varsayılan değer atama mekanizması eklenerek giderilmiştir.

- **Metin Benzerliği Düşük Sonuçlar:**

İlk aşamada yalnızca iş ilanı başlıkları kullanıldığında öneri sonuçlarının yeterince anlamlı olmadığı gözlemlenmiştir. Bu sorun, ilan açıklamaları ve yetenek alanlarının da analiz sürecine dahil edilmesiyle çözülmüştür.

- **Kullanıcı Arayüzü Hataları:**

Bazı kullanıcı etkileşimlerinde boş girişler sistem hatasına yol açmıştır. Girdi doğrulama kontrolleri eklenerek bu problemler ortadan kaldırılmıştır.

Bu hataların tespiti ve giderilmesi süreci, yazılımın olgunlaşmasına ve daha kararlı bir yapıya kavuşmasına katkı sağlamıştır.

7.5. Genel Değerlendirme

Uygulanan testler sonucunda geliştirilen yazılımın fonksiyonel gereksinimleri büyük ölçüde karşıladığı, kullanıcı girdilerine doğru ve tutarlı çıktılar ürettiği gözlemlenmiştir. Test süreci, yazılımın güvenilirliğini artırmış ve sistemin gerçek kullanım senaryolarına hazır hale gelmesini sağlamıştır.

8. YAZILIM KALİTESİ VE GÜVENLİĞİ

Bu bölümde, geliştirilen yazılımın kalite ve güvenlik boyutları ele alınmakta; kodun sürdürülebilirliği, bakım yapılabilirliği ve güvenli bir şekilde çalışabilmesi için benimsenen yaklaşımlar açıklanmaktadır. Yazılımın yalnızca işlevsel olarak doğru çalışması değil, aynı zamanda uzun vadede geliştirilebilir ve güvenilir olması hedeflenmiştir.

8.1. Yazılım Kalitesi Yaklaşımı

Proje kapsamında yazılım kalitesini artırmak amacıyla temiz kod (Clean Code) ve modüler tasarım prensipleri benimsenmiştir. Kodun okunabilirliğini artırmak için anlamlı değişken ve fonksiyon isimleri kullanılmış, tek sorumluluk ilkesine (Single Responsibility Principle) uygun bir yapı oluşturulmuştur. Her modül yalnızca kendi sorumluluk alanına odaklanacak şekilde tasarlanmış, bu sayede kodun anlaşılması ve bakımının kolaylaştırılması hedeflenmiştir.

Ayrıca, tekrar eden kod bloklarından kaçınılmış ve ortak işlevler fonksiyonlar aracılığıyla soyutlanmıştır. Bu yaklaşım, hem hata yapma olasılığını azaltmış hem de yazılımın sürdürülebilirliğine katkı sağlamıştır. Kod yapısı, ileride yeni özelliklerin eklenmesine veya mevcut bileşenlerin değiştirilmesine uygun olacak şekilde esnek tutulmuştur.

8.2. Kalite Metrikleri ve Standartlar

Yazılım kalitesi değerlendirilirken, genel kabul görmüş ISO/IEC 25010 Yazılım Kalite Modeli temel alınmıştır. Bu model doğrultusunda özellikle aşağıdaki kalite özellikleri ön planda tutulmuştur:

- **Bakım yapılabilirlik:** Modüler yapı ve düşük bağımlılık
- **Kullanılabilirlik:** Sade ve anlaşılır kullanıcı arayüzü
- **Güvenilirlik:** Hatalı girdilere karşı dayanıklılık
- **Performans:** Orta ölçekli veri setleri üzerinde kabul edilebilir yanıt süreleri

Kod karmaşıklığı, fonksiyon uzunlukları ve modül bağımlılıkları manuel olarak değerlendirilmiş; fonksiyonların mümkün olduğunca kısa ve anlaşılır tutulmasına özen gösterilmiştir. Bu ölçütler, yazılımın genel kalite seviyesini artırmaya yönelik olarak kullanılmıştır.

8.3. Güvenlik Yaklaşımı

Geliştirilen yazılım, yerel ortamda çalışan bir masaüstü uygulaması olarak tasarlandığından, güvenlik önlemleri bu bağlamda ele alınmıştır. Sistem içerisinde kullanıcıdan alınan veriler yalnızca geçici olarak bellekte tutulmakta ve herhangi bir dış sunucuya gönderilmemektedir. Bu yaklaşım, veri gizliliği açısından önemli bir avantaj sağlamaktadır.

Proje kapsamında kimlik doğrulama veya yetkilendirme gerektiren bir kullanıcı yönetim sistemi bulunmamaktadır. Bunun temel nedeni, uygulamanın bireysel ve yerel kullanım senaryolarına yönelik olarak tasarlanmış olmasıdır. Ancak olası güvenlik risklerine karşı aşağıdaki önlemler alınmıştır:

- Kullanıcı girdileri doğrulanarak boş veya hatalı veri girişleri engellenmiştir
- Dosya yükleme aşamasında beklenen format ve kolon kontrolleri yapılmıştır
- Harici bağlantılar yalnızca kullanıcı onayı ile tarayıcı üzerinden açılmaktadır

Bu önlemler, hatalı kullanım ve beklenmeyen senaryolar karşısında sistemin güvenli ve kararlı kalmasını sağlamaktadır.

8.4. Olası Saldırı Senaryoları ve Önlemler

Yerel masaüstü uygulaması olması nedeniyle sistem; ağ tabanlı saldırılara, yetkisiz erişime veya veri sızıntısına karşı doğal olarak sınırlı bir risk taşımaktadır. Bununla birlikte, hatalı dosya yükleme veya kötü niyetli veri içerikleri gibi senaryolar göz önünde bulundurulmuş ve veri doğrulama mekanizmaları ile bu riskler minimize edilmiştir. Kullanıcıdan gelen girdilerin doğrudan sistem komutlarıyla ilişkilendirilmemesi, olası enjeksiyon risklerini de ortadan kaldırmaktadır.

8.5. Kod Kalite Araçları ve İyileştirmeler

Bu proje kapsamında otomatik kod kalite analiz aracı (örneğin SonarQube) doğrudan kullanılmamıştır. Ancak geliştirme sürecinde kod; okunabilirlik, fonksiyonel ayrışma ve tekrar eden yapılar açısından manuel olarak gözden geçirilmiştir. Gereksiz karmaşıklık oluşturan bölümler sadeleştirilmiş, uzun fonksiyonlar daha küçük parçalara ayrılmıştır.

İleride projenin daha büyük bir ekip tarafından geliştirilmesi veya web tabanlı bir yapıya dönüştürülmesi durumunda, SonarQube gibi araçlar kullanılarak kod kalitesinin metriklerle ölçülmesi ve teknik borcun yönetilmesi mümkün olacaktır.

8.6. Genel Değerlendirme

Benimsenen kalite ve güvenlik yaklaşımları sayesinde geliştirilen yazılım; okunabilir, sürdürülebilir ve güvenilir bir yapıya kavuşmuştur. Kod kalitesine verilen önem, yazılımın uzun vadede geliştirilebilir olmasını sağlarken; güvenlik açısından alınan önlemler, kullanıcı verilerinin korunmasına ve sistemin kararlı çalışmasına katkıda bulunmuştur.

9. SONUÇ VE TARTIŞMA

Bu proje kapsamında, bireylerin yetkinlikleri ve deneyimleri doğrultusunda kendilerine en uygun iş ilanlarını otomatik olarak önerebilen bir yazılım sisteminin geliştirilmesi amaçlanmıştır. Proje başlangıcında belirlenen hedefler doğrultusunda; iş ilanı verilerinin analiz edilmesi, kullanıcı profili ile ilanlar arasında anlamlı eşleştirmeler yapılması ve bu sürecin kullanıcı dostu bir arayüz aracılığıyla sunulması hedeflenmiştir. Geliştirilen yazılımın mevcut iş arama süreçlerindeki manuel filtreleme ve bilgi karmaşası problemlerini azaltma noktasında büyük ölçüde başarılı olduğu görülmektedir.

Elde edilen sonuçlar, sistemin kullanıcıdan alınan metin tabanlı profil bilgileri ile iş ilanları arasında tutarlı ve anlamlı eşleştirmeler yapabildiğini göstermektedir. Özellikle başlık, açıklama ve yetenek alanlarının birlikte analiz edilmesi, öneri kalitesini artırmış ve kullanıcıların kendilerine daha uygun ilanlara daha kısa sürede ulaşabilmesini sağlamıştır. Bu yönüyle yazılım, ele alınan problemin çözümüne pratik ve işlevsel bir katkı sunmuştur.

Proje süreci boyunca çeşitli teknik zorluklarla karşılaşılmıştır. Veri setlerindeki eksik veya tutarsız alanlar, öneri sonuçlarının doğruluğunu olumsuz etkileyen önemli bir faktör olmuştur. Bu sorunlar, veri ön işleme aşamasında yapılan kontroller ve iyileştirmelerle giderilmiştir. Ayrıca metin benzerliği yöntemlerinin ilk aşamada istenilen performansı vermemesi, analiz yaklaşımının yeniden ele alınmasını gerektirmiştir. Bu durum, sistemin daha kapsamlı metin alanlarıyla beslenmesi ve parametre ayarlamaları yapılarak aşılmıştır.

Proje sonucunda elde edilen en önemli kazanımlardan biri, gerçek dünya verileriyle çalışan bir öneri sisteminin uçtan uca tasarlanması ve hayata geçirilmesi olmuştur. Yazılım mühendisliği süreçleri, mimari tasarım, test ve kalite değerlendirme aşamalarının bütüncül bir şekilde uygulanması, geliştiricilere önemli deneyimler kazandırmıştır.

Gelecekte yapılabilecek çalışmalar kapsamında sistemin web tabanlı bir yapıya taşınması, kullanıcı hesapları ve kişiselleştirilmiş geçmiş analizlerinin eklenmesi mümkündür. Ayrıca daha gelişmiş doğal dil işleme teknikleri ve makine öğrenmesi modelleri kullanılarak öneri doğruluğu artırılabilir. Gerçek zamanlı veri çekme, API entegrasyonları ve çoklu platform desteği gibi geliştirmelerle sistemin etki alanı genişletilebilir. Bu doğrultuda proje, geliştirilmeye açık ve ölçeklenebilir bir temel sunmaktadır.

KAYNAKÇA

[1] F. Ricci, L. Rokach and B. Shapira, Recommender Systems Handbook, 2nd ed., Boston, MA, USA: Springer, 2015.

[2] C. D. Manning, P. Raghavan and H. Schütze, Introduction to Information Retrieval, Cambridge, U.K.: Cambridge University Press, 2008.