



Grup No: 10



Eskişehir Osmangazi Üniversitesi 2024-2025 Güz Dönemi

# ÜRETİM VERİ TABANI VERİ YÖNETİM SİSTEMİ

VERİ TABANI YÖNETİM SİSTEMLERİ

Kadriye HARMANCI 152120221094  
Abdullah Taha AYDIN 152120211055  
Elif Suna GEGİN 15212022105  
Umay Ece MANTAR 152120221127

# İçindekiler

**03** Projenin Tanıtımı

**04** Amaç ve Hedef

**05** Projenin Eski ve Güncel Halinin Karşılaştırılması

**06** Projenin Genel Değerlendirmesi

**07-08** Veri Yapısı Tasarımı

**09-10** Kullanılan Teknolojiler

**11-13** Karmaşık Sorgular

**14-16** Saklı Yordamlar

**17-18** Error Handling

**19-20** Data Cleaning-Categorizing

**21** LLM Bağlantısı

**22-23** Arayüz Tasarımı

**24** Demo Gösterim


**25-26** Excel Dosyasının Python'a Aktarımı

**27** Zaman Olsaydı Ekleneceler





# Projenin Tanıtımı

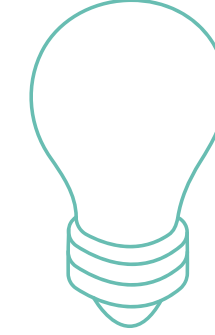
- 
- Projemiz, Ford Otosan'ın üretim bandındaki ürünler ve teknik müdahalelere ilişkin verileri toplayan, düzenleyen ve analiz eden; vektör veritabanı altyapısını baz alan bir uygulamadır.
  - Proje, mevcut veri tabanındaki hataları gidermek ve veri yönetim sistemini Vector DB ve Python teknolojileri kullanarak optimize etmeyi hedeflemektedir. PYQT5 ile geliştirilecek bir masaüstü uygulaması olarak tasarlanmış, çalışanlara daha hızlı ve doğru bilgi erişimi sağlayacak bir sistemdir.





## Amaç

- Ford Otosan'ın üretim hattındaki verileri analiz ederek chatbot performansını değerlendirmek, mevcut eksiklikleri gidermek ve optimize edilmiş bir veritabanı oluşturmaktır.
- Bu sayede süreçlerdeki verimlilik artırılarak teknik müdahaleler daha hızlı ve doğru şekilde gerçekleştirilecektir.



## Hedef

- Vektör veritabanı ile verilerin doğruluğunu artırarak chatbot performansını iyileştirmek ve karar destek mekanizmalarına daha verimli katkı sağlamaktır.
- Böylece, üretim hattındaki sorunlar daha hızlı tespit edilip çözülerek operasyonel süreçlerin kesintisiz ilerlemesi sağlanacaktır.

# Projenin Eski ve Güncel Halinin Karşılaştırılması

- Projede UI için kullanım rahatlığı ve pratikteki kolaylığından dolayı PYQT5 yerine tkinter kütüphanesi kullanılmaya karar verildi.
- ·sentence\_transformers kütüphanesi; Tensorflow API, Sckit-learn ve Numpy 'den daha özel bir NLP odaklı çözüm olduğu için tercih edildi.
- ·openpyxl: ChromaDB'de veri saklama Excel yerine DuckDB ve Parquet üzerinden yapılıyor.
- ·Seaborn ve Matplotlib: Proje veri görselleştirme yerine interaktif sorgu ve yanıtlamaya odaklanmış durumda olduğu için bu kütüphaneler kullanılmıyor.
- ·Arayüzün yönlendirdiği Google hesap açma sayfası ile yetkilendirme üzerinden sisteme erişim sağlanmaktadır.



# Projenin Genel Değerlendirmesi

ChromaDB ve Python ile çalışma deneyimi kazanıldı. UI tasarımı, müşteri beklentilerine uygun olarak geliştirildi ve ekip içi iş birliği ile dokümantasyon sağlandı. Ekip, veri yönetimi ve Excel hakkında pratik bilgi edindi.

ChromaDB, Google Gemini API ve UI entegrasyonu test edilip optimize edildi. Google OAuth ve RAG mimarisi doğrulandı. Sistem, performans ve veri tutarlılığı açısından test edilecek ve kullanıcı beklentileri karşılanacaktır.

Proje, verilerin düzenlenmesiyle operasyonel verimliliği artırdı. Veri tabanı, Ford Otosan'ın ihtiyaçlarına uygun hale getirildi ve sistemin etkin kullanımı sağlandı.



# Veri Kaynağı

- Excel dosyasındaki aşağıdaki sütunlardan veriler, sistem veri tabanına aktarılacaktır:
- Açıklama: Kısa problem özeti.
- Uzun Açıklama: Problem ve çözümüne dair detaylar.
- Konum: Problemin yaşandığı ekipmanın yeri.
- Ekipman Numarası: Ekipmanın benzersiz kimlik numarası.



# Veri Tabanı Yapısı

Sistem, ChromaDB kütüphanesi kullanarak aşağıdaki yapıda bir veri tabanı oluşturur:

Koleksiyon Adı: equipment\_issues

Bu koleksiyon, ekipman arızalarını saklamak için kullanılır. Dokümanlar:

- İçerik: Uzun açıklama, konum ve ekipman numarasından oluşan metin. Örneğin:

"İçerik: Motor çalışmıyor, Konum: A Hattı, Ekipman: EMS256"

- Metadata (Ek bilgiler ve filtreleme için):

Konum: Örneğin, "A Hattı".

Ekipman Numarası: Örneğin, "EMS256".

Açıklama: Problem özeti.

ID: Her kayıt için benzersiz bir kimlik oluşturulur.

Format: Satır indeksine dayalı string (str(index)), vektörler oluşturulurken atanır.

Oluşturulan her vektör 384 boyutludur.





# Kullanılan Teknolojiler

---

1.4

## Flask

- Hafif bir web çerçevesi olup Google OAuth entegrasyonu ve kullanıcı giriş işlemlerini yönetmek için kullanıldı.
- Kullanıcıların tarayıcı üzerinden chatbot'a erişimini sağlar.

## ChromaDB

- Vektör veritabanı olarak kullanıldı; sorguların geçmiş veriyle eşleştirilmesini sağlar.
- Hızlı veri erişimi ve arama işlemleri için tercih edildi.

## Sentence Transformers

- Metinleri sayısal vektörlere dönüştürerek benzerlik hesaplamalarında kullanıldı.
- Kullanıcı sorgularının ve verilerin anlamlı eşleştirilmesini sağlar.



# Kullanılan Teknolojiler

---

## Google Generative AI (Gemini)

- Kullanıcı sorgularına anlamlı ve teknik çözümler üretmek için kullanıldı.
- LLM modeli sayesinde detaylı ve doğru cevaplar oluşturur.

## Tkinter

- Basit bir masaüstü arayüzü sağlamak için kullanıldı.
- Kullanıcıların sorgu girmesi ve cevapları görmesi için grafiksel arayüz oluşturur.

## Pandas

- Veri temizleme, düzenleme ve Excel dosyalarından veri okuma işlemleri için kullanıldı.
- Verinin işlenip analiz edilmesini kolaylaştırır.



# Karmaşık Sorgular

## 1. ChromaDB Sorguları (Veritabanı Erişimi):

Kullanıcının girdiği sorguya karşılık gelen verileri collection.query ile alıyor.

```
results = collection.query(query_texts=[user_query], n_results=2)
```

Bu sorgu, ChromaDB'de kaydedilmiş belgelere en yakın olanları bulmak için vektör tabanlı arama kullanıyor.

## 2. Gemini API Kullanımı (LLM Yanıt Üretimi):

generate\_LLM\_answer fonksiyonu aracılığıyla Google Gemini API'yi çağırarak, kullanıcının sorgusu ve bulunan belge içeriğiyle cevap oluşturuyor.

```
response = chat.send_message(prompt + context)
```



# Karmaşık Sorgular

## 3. Google OAuth İşlemleri (Kimlik Doğrulama):

Kullanıcı girişini sağlamak için Google OAuth kullanılıyor, verilerin doğruluğu kontrol ediliyor

```
token = oauth.google.authorize_access_token()
nonce = session.pop('nonce', None)
if not nonce:
    return "Nonce eksik!", 400
user_data = oauth.google.parse_id_token(token, nonce=nonce)
```

## 4. Vektör Temsili (Sentence Transformer):

Kullanıcıdan alınan sorgunun vektör temsili oluşturuluyor ve veritabanında sorgu için kullanılıyor.

```
query_vector = model.encode(user_query)
```



# ■ Karmaşık Sorgular

## 5. Çoklu İstemci ve Sunucu Arayüzleri (Flask + Tkinter):

Flask, OAuth kimlik doğrulamasını sağlarken, Tkinter üzerinden kullanıcı arayüzü sağlanıyor. İki bileşen arasında veri akışı sağlanıyor.

## 6. Threading (Paralel İşlemler):

Flask sunucusu arka planda çalışırken Tkinter uygulaması aktif tutuluyor:

```
threading.Thread(target=start_flask).start()
```



# Saklı Yordamlar

## 1. handle\_query( ):

Kullanıcı sorgusunu alır, Sentence Transformers ile vektörleştirir, ChromaDB'de arama yapar ve Gemini API kullanarak cevap üretir. Tüm bu adımları tek bir işlemde gerçekleştirir.

```
def handle_query():
    user_query = query_input.get().strip() # Kullanıcı sorgusunu alın ve boşlukları temizleyin
    if not user_query:
        update_response("Lütfen bir sorgu girin.")
        return

    update_response("Cevaplanıyor...") # "Cevaplanıyor..." mesajını göster
    root.update() # GUI'yi güncelle

    try:
        query_vector = model.encode(user_query)
        results = collection.query(query_texts=[user_query], n_results=2)

        if results['documents'][0]:
            context = "\n\n".join(results["documents"][0])
            prompt = f"Sorgu: {user_query}"
            gemini_response = generate_LLM_answer(prompt, context, RAG_LLM)

            response = "CHATBOT YANITI:\n" + gemini_response if gemini_response else "Gemini API'den sonuç alınamadı."
            save_to_xml(user_data['email'], query_input.get(), response) # Sorguyu ve yanıtı XML dosyasına kaydedin
        else:
            prompt = f"Sorgu: {user_query}\n\nDaha önce böyle bir sorun yaşanmadı, bunu belirt ve genel bir çözüm öner."
            gemini_response = generate_LLM_answer(prompt, "", RAG_LLM)

            response = "ChromaDB'den sonuç bulunamadı.\n" + (gemini_response if gemini_response else "Gemini API'den sonuç alınamadı.")
            save_to_xml(user_data['email'], query_input.get(), response) # Sorguyu ve yanıtı XML dosyasına kaydedin
    except Exception as e:
        response = f"Hata oluştu: {str(e)}"

    update_response(response)
```

# Saklı Yordamlar

## 2- LLM Özelleştirmesi

```
def build_chatBot(system_instruction):  
    model = genai.GenerativeModel('gemini-1.5-flash-latest', system_instruction=system_instruction)  
    chat = model.start_chat(history=[])  
    return chat  
  
def generate_LLM_answer(prompt, context, chat):  
    response = chat.send_message(prompt + context)  
    return response.text  
  
def generateRAG_LLM(prompt):  
    RAG_LLM = build_chatBot(system_prompt)  
    return RAG_LLM  
  
system_prompt= """ You are a technical support assistant.  
  
Your primary role is to provide concise, actionable solutions to technical problems based on provided context and historical problem records. Here's how to respond:  
  
1. You will receive a user query alongside two historical problem records that are similar to the current issue. Analyze the query and the records to provide a solution.  
2. Based on the information provided, deliver a response in bullet points with clear steps (maximum 5-6 points) on how the problem can be solved.  
3. If no historical problem records are provided, or if the query does not match any known issues, respond with: "Bu sorunla daha önce hiç karşılaşılmamış. Lütfen daha fazla bilgi veriniz."  
4. Ensure responses are concise and fully in Turkish.  
5. In some cases, you might be asked questions about the chat session itself (e.g., summarizing, listing questions). For these, do not refer to the user query or the context provided.
```



# ■ Saklı Yordamlar

## 3. `assign_category( )`:

Veriyi analiz eder, açıklamaların kategori benzerliklerini ölçer ve en uygun kategoriye otomatik olarak atar. Tekrarlanabilir ve sistematik bir sınıflandırma sağlar.

```
def assign_category(row):  
    similarities = {cat: cosine_similarity([row['aciklama_vector']], [vec]).item() for cat, vec in category_vectors.items()}  
    return max(similarities, key=similarities.get) #
```

## 4. `generate_LLM_answer()`:

Kullanıcı sorgusu ve bağlamı kullanarak Gemini API üzerinden anlamlı ve teknik çözümler üretir. Tekrarlayan cevap oluşturma işlemini tek bir yapıda yönetir.

```
def generate_LLM_answer(prompt, context, chat):  
    response = chat.send_message( prompt + context)  
    return response.text
```





# Error Handling

## Kullanıcı Sorguları ve Hata Kontrolü

Kullanıcıdan gelen sorguların doğru şekilde işlenebilmesi için, geçersiz veya boş sorgulara karşı bir hata kontrolü yapılmıştır. Eğer kullanıcı boş bir sorgu gönderirse, sistem bir hata mesajı verir ve işlem yapılmaz.

```
def handle_query():  
    user_query = query_input.get().strip()  
    if not user_query:  
        update_response("Lütfen bir sorgu girin.")  
        return
```



# Error Handling

## Google OAuth Giriş Hatası (Flask ile Entegre)

Google OAuth entegrasyonu sırasında, kullanıcı kimlik doğrulama ve erişim hataları için hata yönetimi yapılmaktadır. Eğer bir hata oluşursa, hata mesajı kullanıcıya gösterilir.

```
@app.route('/authorize')
def authorize():
    global user_data
    try:
        token = oauth.google.authorize_access_token()
        nonce = session.pop('nonce', None)
        if not nonce:
            return "Nonce eksik!", 400
        user_data = oauth.google.parse_id_token(token, nonce=nonce)
        session['user'] = user_data
        return redirect('/success')
    except Exception as e:
        return f"Bir hata oluştu: {e}", 500
```

# Data Cleaning-Categorizing

## Veri kategorizasyonu

Mevcut verilerde, açıklamalar sütununda yazım hataları veya benzeri nedenlerle kategori tekrarları bulunmaktadır. Ayrıca, genel bir kategorilendirme sütununun eksikliği, verilerin düzenlenmesini ve analiz edilmesini zorlaştırmaktadır.

- ☒ (Tümünü Seç)
- ☒ atc arızası
- ☒ ATC ARIZASI
- ☒ atc arızası takım değiştirmiyor
- ☒ atc arızası tezgah baslatılamıyor
- ☒ atc kapağı çıktı
- ☒ ATC KAPAĞI KIRIK
- ☒ Atc kapı arızası.
- ☒ atc kapı switch arızası

Eski ve hatalı açıklamalar sütunu

- ☒ (Tümünü Seç)
- ☒ Atc Sorunları
- ☒ Diğer
- ☒ Kapı Sorunları
- ☒ Motor Sorunları
- ☒ Takım Sorunları

Yeni kategori sütunu

- ☒ atc arızası
- ☒ atc arızası takım değiştirmiyor
- ☒ atc arızası tezgah baslatılamıyor
- ☒ atc kapağı çıktı
- ☒ atc kapağı kırık
- ☒ atc kapı arızası.
- ☒ atc kapı switch arızası
- ☒ atc kapı switch arızası

Düzenlenmiş açıklamalar

# ■ Data Cleaning-Categorizing

Bu işlemde, veri temizleme adımında metinlerdeki yazım hataları düzeltilir, harfler küçültülür ve metinler standart bir forma getirilir. Temizlenen veriler, kategorize etme işlemleri için sonraki adımlarda kullanıma hazır hale getirilir.

```
def clean_text(text):  
    text = text.replace("İ", "i")  
    text = re.sub(r'arizasi', 'arızası', text, flags=re.IGNORECASE)  
    text = text.lower()  
    return text
```

```
# Temizleme işlemi  
data['Açıklama'] = data['Açıklama'].apply(clean_text)  
  
# SentenceTransformer modeli yükleme  
model = SentenceTransformer('all-MiniLM-L6-v2')  
  
# Açıklamaları vektöre dönüştürme  
data['aciklama_vector'] = data['Açıklama'].apply(model.encode)
```

# ■ LLM Bağlantısı

3.3.1

Gemini (gemini-1.5-flash-latest) adlı LLM modeline API üzerinden erişim sağlıyoruz. Eğer kullanıcının yaşadığı problem daha önce veri tabanında kayıtlıysa, bu veriler veri tabanından alınarak LLM'e gönderilir ve model bu bilgilere dayanarak bir çözüm üretir. Eğer problem veri tabanında bulunmuyorsa, Gemini problemin daha önce yaşanmadığını belirtir ve kullanıcının bir yetkiliyle iletişime geçmesini önerir.

```
try:
    query_vector = model.encode(user_query)
    results = collection.query(query_texts=[user_query], n_results=2)

    if results['documents'][0]:
        context = "\n\n".join(results["documents"][0])
        prompt = f"Sorgu: {user_query}"
        gemini_response = generate_LLM_answer(prompt, context, RAG_LLM)

        response = "CHATBOT YANITI:\n" + gemini_response if gemini_response else "Gemini API'den sonuç alınamadı."
        save_to_xml(user_data['email'], query_input.get(), response) # Sorguyu ve yanıtı XML dosyasına kaydedin
    else:
        prompt = f"Sorgu: {user_query}\n\nDaha önce böyle bir sorun yaşanmadı, bunu belirt ve genel bir çözüm öner."
        gemini_response = generate_LLM_answer(prompt, "", RAG_LLM)

        response = "ChromaDB'den sonuç bulunamadı.\n" + (gemini_response if gemini_response else "Gemini API'den sonuç alınamadı.")
        save_to_xml(user_data['email'], query_input.get(), response) # Sorguyu ve yanıtı XML dosyasına kaydedin
except Exception as e:
    response = f"Hata oluştu: {str(e)}"
```

Ford Otosan Akıllı Bakım Asistanı

Ford Otosan Akıllı Bakım Asistanı

Not: Ford Otosan Akıllı Bakım Asistanı geliştirme aşamasında olan bir çalışmadır.  
Giriş Yapan Kullanıcı: taha5437@gmail.com

Sorgunuzu buraya yazınız:

atc kolu yok ne yapmalıyım

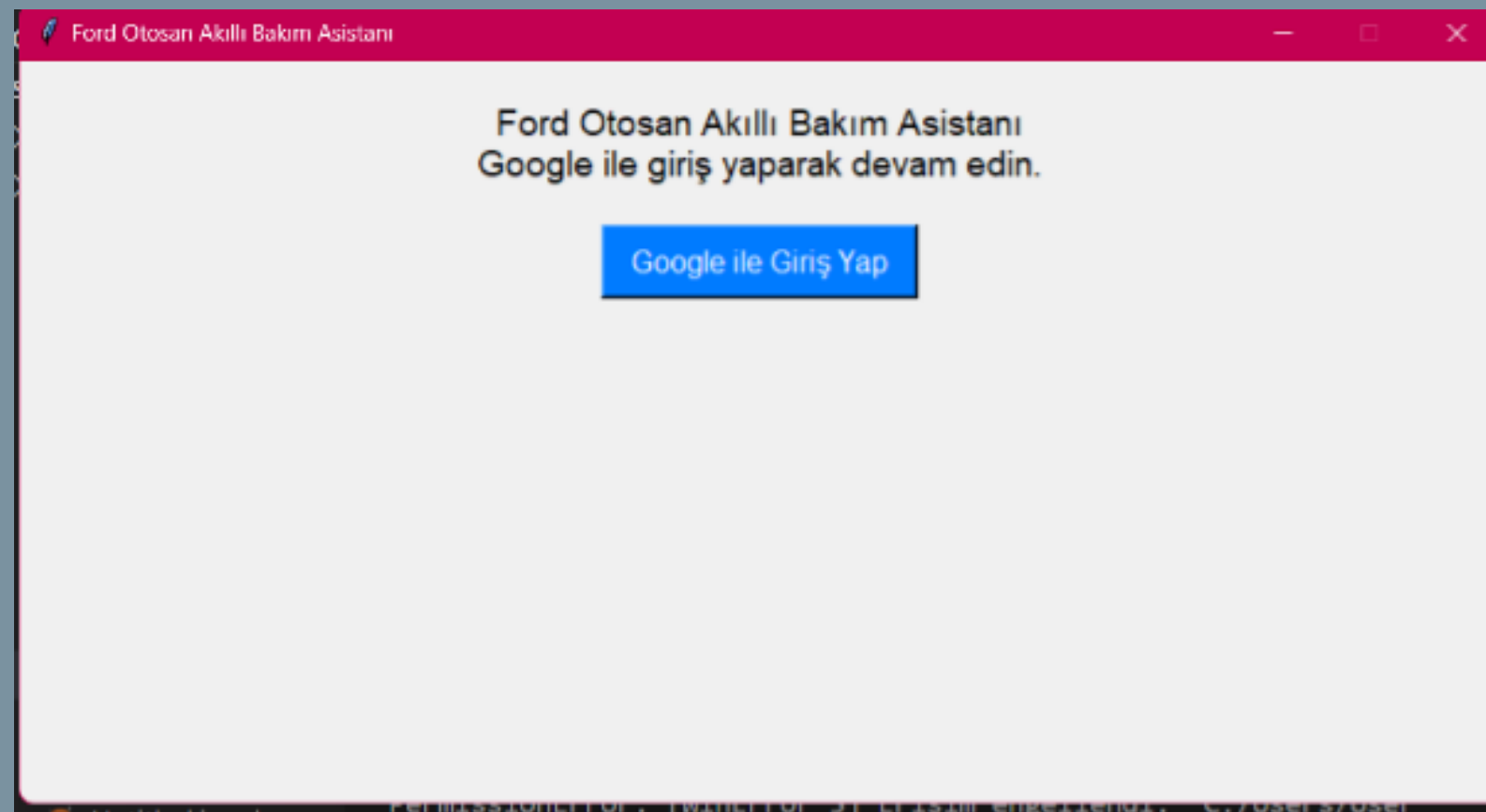
Gönder Geçmiş Geçmiş Sil

CHATBOT YANITI:  
Bu sorunla daha önce hiç karşılaşmamış. Lütfen yetkili teknik destek sorumlusuna ulaşınız.

# Arayüz Tasarımı

# Giriş Ekranı

- Google OAuth kullanarak kullanıcı girişini sağlar.
- Kullanıcı giriş yaptıktan sonra chatbot ekranına geçiş yapar.





# Arayüz Tasarımı

## Chatbot Ekranı

- Sorgu Girişi: Kullanıcının yazdığı soruları alır.
- Gönder Butonu: Sorguyu işlemesi için arka uca iletir.
- Cevap Alanı: Gelen yanıtları kullanıcıya gösterir.
- Geçmiş Sorgulara Erişim: Kullanıcılar, daha önce verdikleri sorgulara ve alınan yanıtlara kolayca ulaşabilirler.

Sorgu: atc kolu arızalı

Yanıt: CHATBOT YANITI:

- \* ATC kolunun fiziksel hasarını inceleyin.
- \* Kolda gevşek kablo veya bağlantı olup olmadığını kontrol edin.
- \* ATC kolunun manuel olarak doğru konumda olup olmadığını doğrulayın.
- \* Referans değerlerini tekrar alın ve değerlerin normal olup olmadığını kontrol edin.
- \* Gerekirse, ATC kolunu değiştirin veya onarım için yetkili servise gönderin.
- \* Sorun devam ederse, sistemin diğer bileşenlerini de kontrol edin.

Sorgu: makineye su gelmiyor

Yanıt: CHATBOT YANITI:

- \* Su besleme hattındaki vanaların açık olduğundan emin olun.
- \* Su basıncını kontrol edin. Basınç düşüklüğü varsa, su kaynağını kontrol edin.
- \* Taşlama valfinin doğru şekilde çalıştığından emin olun. Gerekirse tekrar yağlayın veya değiştirin.
- \* Su filtrelerinin tıkalı olup olmadığını kontrol edin ve gerekirse temizleyin veya değiştirin.
- \* Sorun devam ederse, su borularında tıkanıklık olup olmadığını kontrol edin.
- \* Tüm kontrollerden sonra sorun devam ederse, yetkili servisten yardım alın.

### Ford Otosan Akıllı Bakım Asistanı

Not: Ford Otosan Akıllı Bakım Asistanı geliştirme aşamasında olan bir çalışmadır.

Giriş Yapan Kullanıcı: taha5437@gmail.com

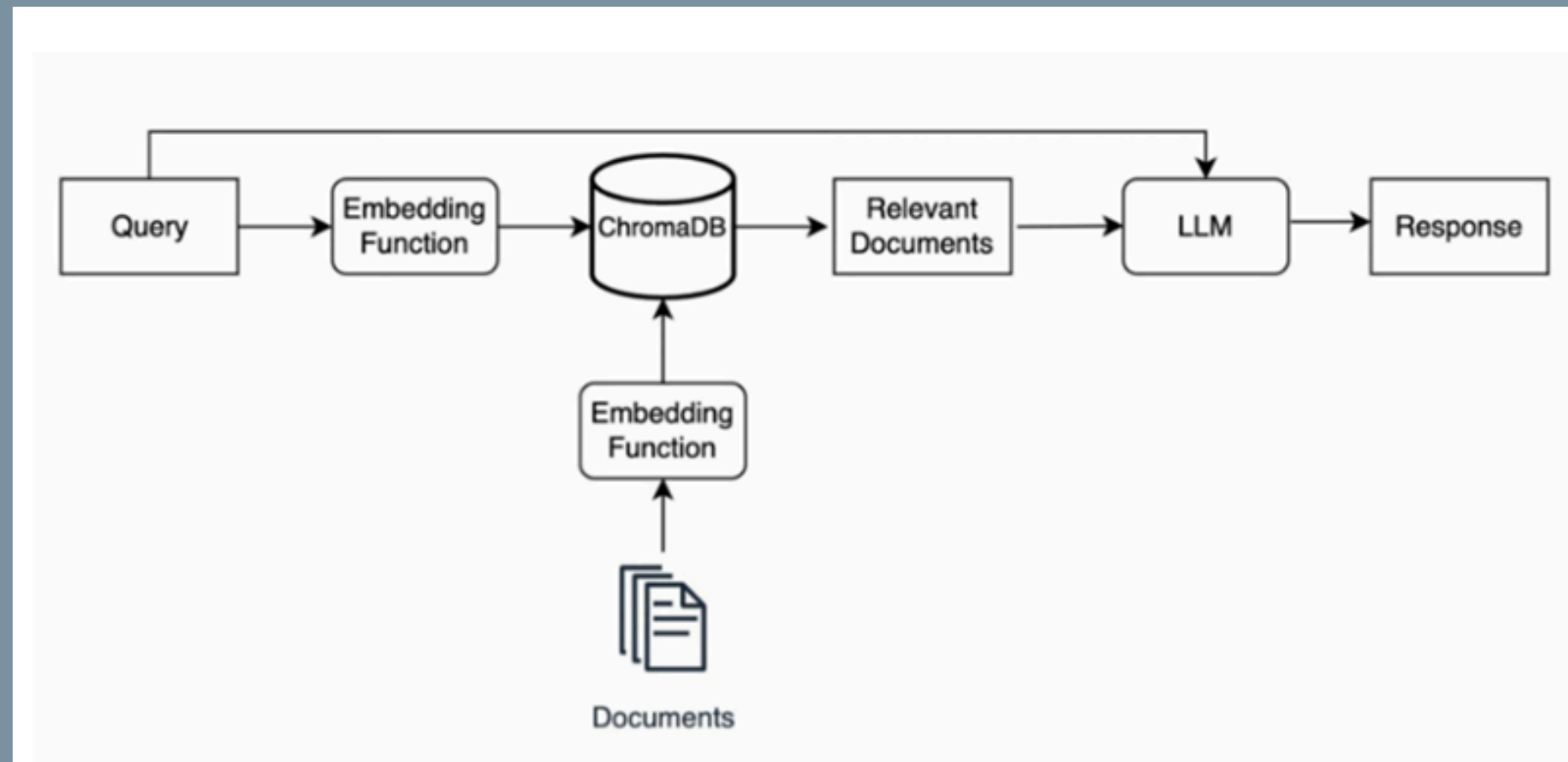
Sorgunuzu buraya yazınız:

Gönder Geçmiş Geçmiş Sil

CHATBOT YANITI:

- \* Su besleme hattındaki vanaların açık olduğundan emin olun.
- \* Su basıncını kontrol edin. Basınç düşüklüğü varsa, su kaynağını kontrol edin.
- \* Taşlama valfinin doğru şekilde çalıştığından emin olun. Gerekirse tekrar yağlayın veya değiştirin.
- \* Su filtrelerinin tıkalı olup olmadığını kontrol edin ve gerekirse temizleyin veya değiştirin.
- \* Sorun devam ederse, su borularında tıkanıklık olup olmadığını kontrol edin.
- \* Tüm kontrollerden sonra sorun devam ederse, yetkili servisten yardım alın.

# DEMO GÖSTERİM





# Excel Dosyasını Python'a Aktarma

4.1.5

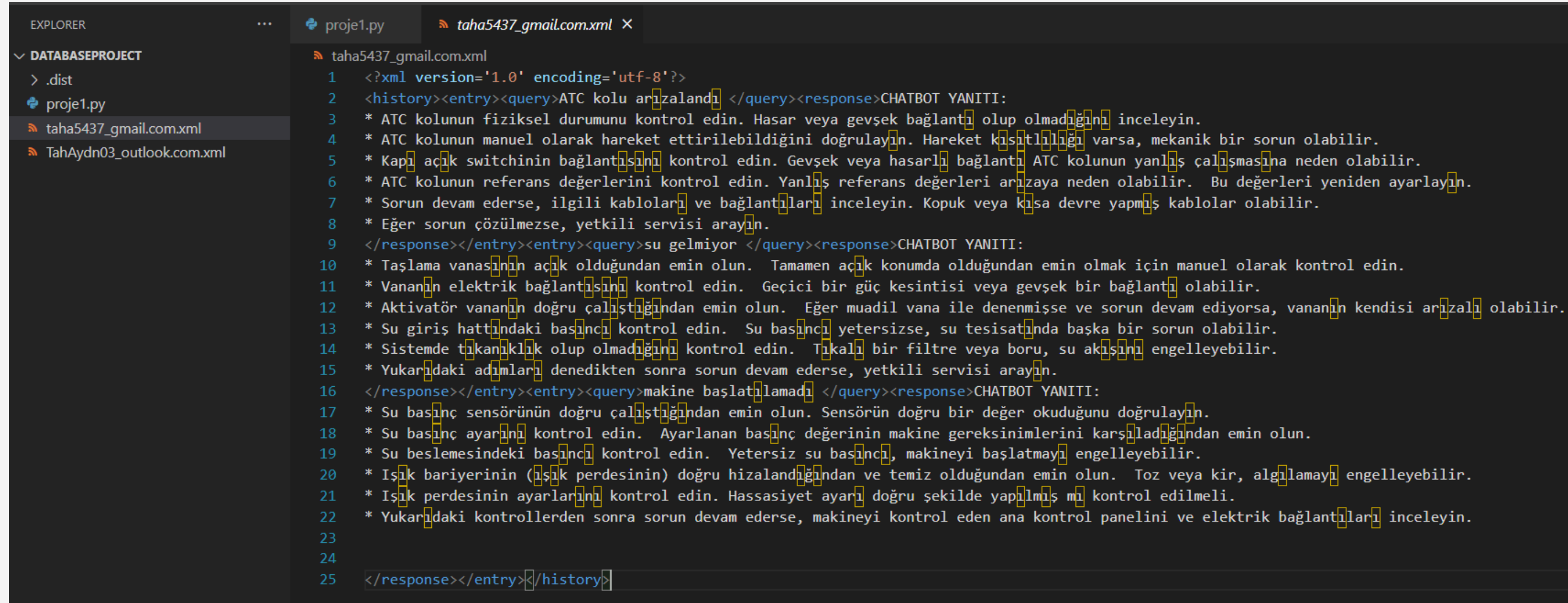
- Excel dosyasının yolu belirtilir
- Dosya yolu, Excel dosyasının bilgisayardaki tam konumudur.
- Pandas kütüphanesi kullanılır
- Pandas, Excel dosyalarını okumak için read\_excel() fonksiyonunu sağlar.

```
1 import pandas as pd
2 from sentence_transformers import SentenceTransformer
3 import chromadb
4 from chromadb.config import Settings
5
6 client = chromadb.PersistentClient(path="C:/Users/User/Desktop/Database/chroma_db")
7 collection_name = "equipment_issues"
8 collection = client.get_or_create_collection(name=collection_name)
9 file_path = 'C:/Users/User/Desktop/Deneme/ÖrnekVeri.xlsx' #Verilerin bulunduğu excel dosyasının dizin yolunu gir.
10 data = pd.read_excel(file_path)
11
12 model = SentenceTransformer('all-MiniLM-L6-v2')
13
14 # 7. Vektörlere Dönüştür ve Depola
15 for index, row in data.iterrows():
16     combined_text = f"{row['Uzun Açıklama']}"
17     vector = model.encode(combined_text)
18     collection.add(
19         documents=[combined_text],
20         metadatas=[{
21             "Konum": row["Konum"],
22             "Ekipman Numarası": row["Ekipman Numarası"],
23             "Açıklama": row["Açıklama"]
24         }],
25         ids=[str(index)]
26     )
27
28 print(f"Koleksiyonda toplam {collection.count()} kayıt var.")
```

# Diğer Entegrasyonlar (XML, XSLSX)

4.1.4

- Kullanıcının chatbot ile olan sohbet geçmişi XML formatında saklanır. Chatbot'un çalıştırıldığı klasörde kaydedilir. Kullanıcı “Geçmiş Sil” butonuna bastığında bu dosya silinerek geçmiş temizlenir.



```
EXPLORER
...
proje1.py
taha5437_gmail.com.xml X
taha5437_gmail.com.xml
1 <?xml version='1.0' encoding='utf-8'?>
2 <history><entry><query>ATC kolu arızalandı </query><response>CHATBOT YANITI:
3 * ATC kolunun fiziksel durumunu kontrol edin. Hasar veya gevşek bağlantı olup olmadığını inceleyin.
4 * ATC kolunun manuel olarak hareket ettirilebildiğini doğrulayın. Hareket kısıtlılığı varsa, mekanik bir sorun olabilir.
5 * Kapı açık switchinin bağlantısını kontrol edin. Gevşek veya hasarlı bağlantı ATC kolunun yanlış çalışmasına neden olabilir.
6 * ATC kolunun referans değerlerini kontrol edin. Yanlış referans değerleri arızaya neden olabilir. Bu değerleri yeniden ayarlayın.
7 * Sorun devam ederse, ilgili kabloları ve bağlantıları inceleyin. Kopuk veya kısa devre yapmış kablolar olabilir.
8 * Eğer sorun çözülmezse, yetkili servisi arayın.
9 </response></entry><entry><query>su gelmiyor </query><response>CHATBOT YANITI:
10 * Taşlama vanasının açık olduğundan emin olun. Tamamen açık konumda olduğundan emin olmak için manuel olarak kontrol edin.
11 * Vananın elektrik bağlantısını kontrol edin. Geçici bir güç kesintisi veya gevşek bir bağlantı olabilir.
12 * Aktivatör vananın doğru çalıştığından emin olun. Eğer muadil vana ile denenmişse ve sorun devam ediyorsa, vananın kendisi arızalı olabilir.
13 * Su giriş hattındaki basınç kontrol edin. Su basıncı yetersizse, su tesisatında başka bir sorun olabilir.
14 * Sistemde tıkanıklık olup olmadığını kontrol edin. Tıkalı bir filtre veya boru, su akışını engelleyebilir.
15 * Yukarıdaki adımları denedikten sonra sorun devam ederse, yetkili servisi arayın.
16 </response></entry><entry><query>makine başlatılamadı </query><response>CHATBOT YANITI:
17 * Su basınç sensörünün doğru çalıştığından emin olun. Sensörün doğru bir değer okuduğunu doğrulayın.
18 * Su basınç ayarlarını kontrol edin. Ayarlanan basınç değerinin makine gereksinimlerini karşıladığından emin olun.
19 * Su beslemesindeki basınç kontrol edin. Yetersiz su basıncı, makineyi başlatmayı engelleyebilir.
20 * Işık bariyerinin (ışık perdesinin) doğru hizalandığından ve temiz olduğundan emin olun. Toz veya kir, algılamayı engelleyebilir.
21 * Işık perdesinin ayarlarını kontrol edin. Hassasiyet ayarı doğru şekilde yapılmış mı kontrol edilmeli.
22 * Yukarıdaki kontrollerden sonra sorun devam ederse, makineyi kontrol eden ana kontrol panelini ve elektrik bağlantılarını inceleyin.
23
24
25 </response></entry></history>
```

# ilerde Yapılması Planlananlar

- Her kullanıcının geçmiş sohbetleri kullanıcının mail adresiyle aynı isimdeki xml dosyasında tutulur. Geçmiş sohbetlere şuan her kullanıcı xml dosyasını açarak ulaşabilir. Bunu şifrelenmiş bir şekilde tutarak diğer kullanıcıların başka kullanıcıların geçmiş sohbetlerinin bulunduğu xml dosyasına erişiminin engellenmesi planlanmaktadır.
- Chatbot'a şuan için sadece google hesabı ile giriş yapılabilmektedir. Bir kayıt oluşturma sistemi yoktur. Chatbot'a bir e-mail adresi veya telefon numarasıyla kayıt olabilme özelliği eklenmesi planlanmaktadır



# TEŞEKKÜRLER