ESKİŞEHİR OSMANGAZİ ÜNİVERSİTESİ
MÜHENDİSLİK-MİMARLIK FAKÜLTESİ
BİLGİSAYAR MÜHENDİSLİĞİ

2024-2025 GÜZ DÖNEMİ

VERİ TABANI YÖNETİM SİSTEMLERİ – A

ÜRETİM BANDI VERİ YÖNETİM SİSTEMİ

GRUP NO: 10

Kadriye HARMANCI 152120221094

Abdullah Taha AYDIN 152120211055

Elif Suna GEGİN 15212022105

Umay Ece MANTAR 152120221127

Firma Yetkilisi, Ömer Ersoy ALANYALI 05326242237

İletişim Bilgileri: ealanyal@ford.com.tr

FORD OTOSAN

Aralık 2024

İçindekiler

1. Giriş

1.1. Projenin Amacı ve Tanımlanması

Bu projede, Ford Otosan'ın üretim bandındaki ürünler ve teknik müdahalelerle ilgili verilerin toplanması, düzenlenmesi ve analiz edilmesi amaçlanmaktadır. Ford Otosan'ın üretim bandında meydana gelen arızalar veya müdahaleler, şirketin teknik birimi tarafından veri tabanına kaydedilmiştir. Bu veri tabanına dayalı olarak, çalışanların ürün veya müdahalelerle ilgili sorgulamalar yapabilmesi için bir yapay zeka tabanlı Chatbot geliştirilmiştir. Chatbot, veri tabanından doğru ve hızlı şekilde bilgi çekerek üretim ve müdahale süreçlerinde çalışanlara destek olmak üzere tasarlanmıştır.

Ancak mevcut veri tabanında bazı hatalar ve eksik kayıtlar bulunmaktadır. Bu durum, Chatbot'un yanlış veya eksik bilgiler sunmasına, yani "Chatbot halüsinasyonu" olarak tabir edilen hatalı cevaplar üretmesine neden olmaktadır.

Bu hataların giderilmesi, verilerin temizlenmesi ve Chatbot performansının artırılması amacıyla veri tabanı tasarımı ve yapılandırma süreci gerçekleştirilecektir.

Şirketle yapılan gizlilik sözleşmesi çerçevesinde, veri hukuku standartlarına uygun olarak veriler temizlenecek, veri tabanı düzenlenecek ve Chatbot'un optimizasyonu sağlanacaktır.

1.2. Projeyi Seçme Amacımız ve Sebebimiz

Bu projeyi seçme amacımız, Ford Otosan'ın mevcut veri yönetim sistemindeki eksikliklerin, şirketin iş süreçlerine olumsuz etkisini gözlemlememiz ve bu sorunları çözme potansiyeline sahip modern veri tabanı teknolojileriyle işletmeye katkı sağlama isteğimizdir. Proje bize hem teknik bir gelişim fırsatı sunmakta hem de gerçek bir iş problemini çözme odaklı bir yaklaşım içermektedir.

1.3. Hedef ve Başarı Kriterlerimiz

Vektör veri tabanı, LLM ve RAG kullanarak yenilikçi çözümlerle var olan problemlerin üstesinden gelerek, şirketin dijital dönüşümüne katkıda bulunmayı ve daha dinamik, güvenilir bir veri altyapısı oluşturmayı hedefliyoruz.

Bu nedenle, proje hem teknik bir gelişim fırsatı sunmakta hem de gerçek bir iş problemini çözme odaklı bir yaklaşım içermektedir.

1.4. Projede Kullanılacak Teknolojiler

Verileri tutmak için Chroma Vector DB, backend ve frontend servisi Python'ın "tkinter, chromadb, sentence_transformers, google.generativeai, flask ,threading webbrowser, secrets, pandas, sklearn.metrics.pairwise import cosine_similarity, re" kütüphanelerini kullanacağız.

1.5. Yapılan Çalışmalar

ChromaDB ile veri tabanı entegrasyonu, Google Gemini API ile yanıt oluşturma süreçleri ve kullanıcı arayüzü (UI) arasındaki veri akışı test edilip optimize edilmiştir. Google OAuth ile kimlik doğrulama devreye alınarak güvenli kullanıcı girişi sağlanmıştır. ,

RAG mimarisi doğrultusunda sistemin kullanıcı sorgularını doğru işlediği doğrulanmış, performans, hata yönetimi ve veri tutarlılığı farklı senaryolar altında test edilerek tüm bileşenlerin uyumlu ve sorunsuz çalışması sağlanmıştır.

1.6. Gelinen Nokta

Belirlenen gereksinimlerin hepsi karşılanmış olup chatbot ile veri tabanı entegrasyonu başarılı bir şekilde tamamlanmıştır.

1.7. Eklenen Yeni Gereksinimler

- Kullanıcı, sisteme Google hesabı üzerinden mail ile oturum açma özelliği kullanarak giriş yapar. Kimlik doğrulama işlemleri Google OAuth ile gerçekleştirilir ve alınan verilerin doğruluğu kontrol edilir.

- Kullanıcı tarafından girilen sorgu, kullanıcı arayüzü (Tkinter) aracılığıyla sisteme gönderilir ve Flask tabanlı sunucu üzerinden işlenir.

- Kullanıcının sorgusu, Sentence Transformers modeli kullanılarak vektör temsiline (embedding) dönüştürülür. Bu vektör boyutu 384 olarak belirlenmiştir.
- Vektör temsili, ChromaDB vektör veritabanında sorgulama yapmak için kullanılır. Bu sorgulama, kullanıcının girdisine en yakın belgelerin getirilmesini sağlar.
- Kullanıcının sorgusuna uygun veri bulunmaması durumunda, sistemin geri bildirim verebilmesi için bir Error Handling mekanizması eklenmiştir.
    - Kullanıcıdan gelen sorgunun boş ya da hatalı olması durumunda, sistem sorgunun doğruluğunu kontrol ederek uygun bir hata mesajı ("Lütfen geçerli bir sorgu girin") verecektir.
    - Kullanıcıya sistemin beklenen davranışı ve hatayı anlamasına yardımcı olacak açıklamalar sunulacaktır (örneğin: "Sorun devam ederse yetkili servisi arayın." veya “Sorun devam ederse, detaylı bir hata kodu kaydı oluşturun ve teknik desteğe başvurun.”).
- Yanıt oluşturma işlemi için eğitilmiş bir dil modeli olan Google Gemini LLM kullanılır. Bu LLM, kullanıcının sorgusunu ve getirilen belge içeriğini baz alarak cevap oluşturur.
- İleride farklı veri tabanlarının bağlanabilmesi için veri ayıklama, temizleme ve kategorize etme fonksiyonları eklenmiştir. Bu fonksiyonlar Trigger mekanizması ile otomatik olarak çalıştırılır.
- Flask tabanlı sunucu, arka planda çalışırken Tkinter kullanıcı arayüzü aktif tutulur. Bu işlem Threading kullanılarak paralel çalışmayı sağlar.
- Kullanıcının sorgu-yanıt döngüsünü açıklayan Retrieval-Augmented Generation (RAG) şeması, sistemin çalışma şeklini ve veri akışını tanımlar. Bu yapı, LLM ve vektör veritabanı entegrasyonunu detaylandırır.
- Sistem, kullanıcı sorgusunu işlerken vektör tabanlı arama, belge getirme, temizleme ve yanıt oluşturma aşamalarını ardışık olarak gerçekleştirir.
    - Açıklamalardaki hatalı yazımlar düzeltildikten ve her biri vektörleştirildikten sonra düzeltilmiş yeni hali ayrı bir dosya olarak mevcut klasörde oluşturulur ve kaydedilir.
- Kullanıcı ve sistem arasındaki veri akışı, Flask ve Tkinter arasındaki entegrasyon ile sağlanır. Bu sayede kimlik doğrulama, sorgu gönderme ve yanıt görüntüleme işlemleri sorunsuz çalışır.
- Arayüzde kullanıcının geçmiş sorguları görebilmesi amacıyla “Geçmiş” butonu, sorgu geçmişini temizlemesi amacıyla “Geçmişi Temizle” butonu eklenmiştir.

1.8. 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.

2. Veri yapısı tasarımı

2.1. Veri Kaynağı

Excel dosyasında bulunan sütunlardan veri tabanında tutulacak verilerin sütunları aşağıdaki gibidir:

- **Açıklama:** Kısa bir problem özeti.
- **Uzun Açıklama:** Problemle ve problemin çözümüyle ilgili detaylı açıklama.
- **Konum:** Problemin meydana geldiği ekipmanın konumu.
- **Ekipman Numarası:** Problemin ilgili olduğu ekipmanın benzersiz kimlik numarası.

2.2. Veri Tabanı Yapısı

Sistem, Chroma kütüphanesini kullanarak aşağıdaki yapı ile bir veri tabanı oluşturur:

- Koleksiyon Adı: “equipment_issues”
  - Bu koleksiyon, ekipman arızalarıyla ilgili verileri tutar.
- Dokümanlar (documents):
  - Depolanan metin verisi, açıklamalar ve ekipman bilgilerini içerir.
  - Format: [Uzun Açıklama] Konum: [Konum] Ekipman: [Ekipman Numarası]
- Metadata (metadatas):
  - Ek bilgiler metadata olarak saklanır ve sorgu sırasında filtreleme yapılmasını sağlar.
  - Alanlar:
    - Konum: Örneğin “A Hattı".
    - Ekipman Numarası: Örneğin, "EMS256".
    - Açıklama: Problem özeti.
- ID:
  - Her kayıt için benzersiz bir kimlik numarasıdır.
  - Format: Satır indeksine dayalı string (str(index)). Vectorler oluşturulurken otomatik olarak atanır.

2.3. Sorgular

Bu projede karmaşık sorgular ve işlemler şunlardır:

ChromaDB Sorguları (Veritabanı Erişimi):

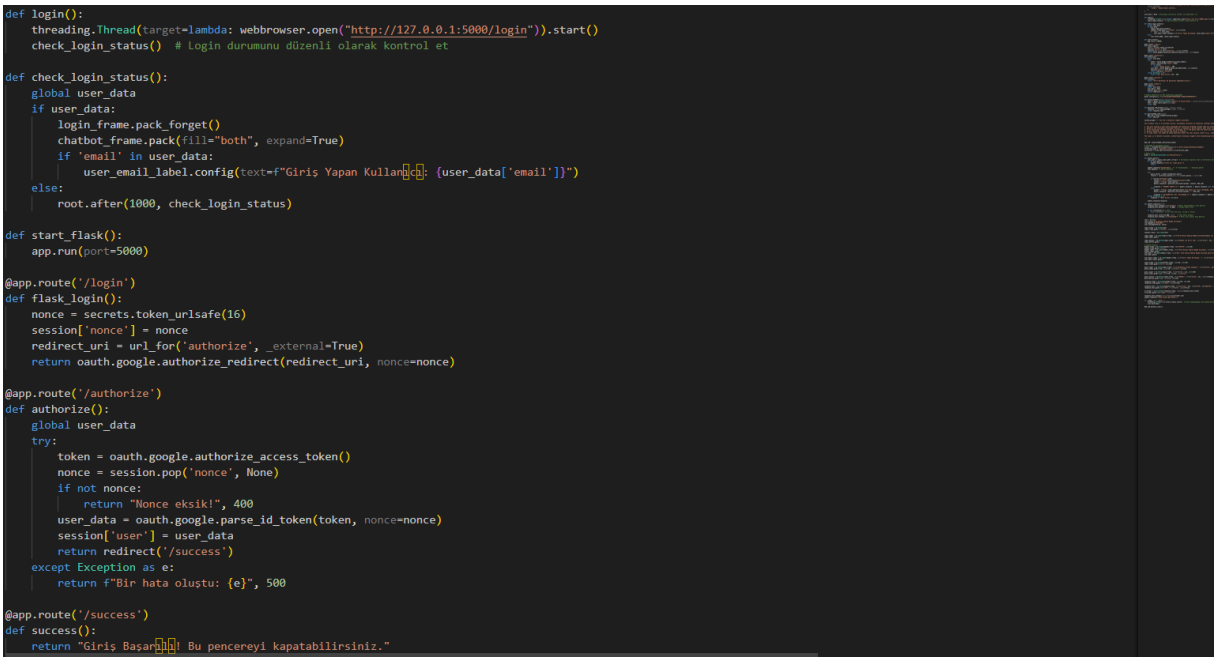
Kullanıcının girdiği sorguya karşılık gelen verileri collection.query ile alıyor.

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

Gemini API Kullanımı (LLM Yanıt Üretimi):

```
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."
        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.")
    except Exception as e:
        response = f"Hata oluştu: {str(e)}"

    update_response(response)
```

Google OAuth İşlemleri (Kimlik Doğrulama):



```
def login():
    threading.Thread(target=lambda: webbrowser.open("http://127.0.0.1:5000/login")).start()
    check_login_status()  # Login durumunu düzenli olarak kontrol et

def check_login_status():
    global user_data
    if user_data:
        login_frame.pack_forget()
        chatbot_frame.pack(fill="both", expand=True)
        if 'email' in user_data:
            user_email_label.config(text=f"Giriş Yapan Kullanıcı: {user_data['email']}")
    else:
        root.after(1000, check_login_status)

def start_flask():
    app.run(port=5000)

@app.route('/login')
def flask_login():
    nonce = secrets.token_urlsafe(16)
    session['nonce'] = nonce
    redirect_uri = url_for('authorize', _external=True)
    return oauth.google.authorize_redirect(redirect_uri, nonce=nonce)

@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

@app.route('/success')
def success():
    return "Giriş Başarılı! Bu pencereyi kapatabilirsiniz."
```

Vektör Temsili (Sentence Transformer):

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

Threading (Paralel İşlemler):

```
if __name__ == '__main__':
    threading.Thread(target=start_flask).start()
    root.mainloop()

RAG_LLM.history.clear()
```

2.4. Saklı Yordamlar

2.4.1. Sorgunun alınıp işlenmesi

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.

```python
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."
        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.")
    except Exception as e:
        response = f"Hata oluştu: {str(e)}"

    update_response(response)
```

2.4.2. LLM Özelleştirmesi

• **build_chatBot(system_instruction):** Bu işlev, özel bir sistem talimatıyla LLM modelini başlatır ve sohbet geçmişi ile özelleştirilmiş bir sohbet oturumu oluşturur.

• **generate_LLM_answer(prompt, context, chat):** Kullanıcı sorgusu ve bağlama dayalı olarak LLM'nin yanıt üretmesini sağlar. Yanıtlar, bağlamla birleştirilerek daha etkili bir sonuç oluşturulur.

• **generateRAG_LLM(prompt):** RAG (Retrieval-Augmented Generation) modelini yapılandırarak belirli bir sistem talimatına uygun bir LLM oluşturur.

```python
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 yetkili tek
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 historical r

Your goal is to deliver accurate, context-aware technical support while maintaining brevity and clarity.

"""
```

2.4.3. Kategorilerin Atanması

- **assign_category( ):** Veriyi analiz eder, açıklamaların kategori benzerliklerini ölçer ve en uygun kategoriye otomatik olarak atar. Tekrarlanabilir ve sistematik bir sınıflandırma sağlar.

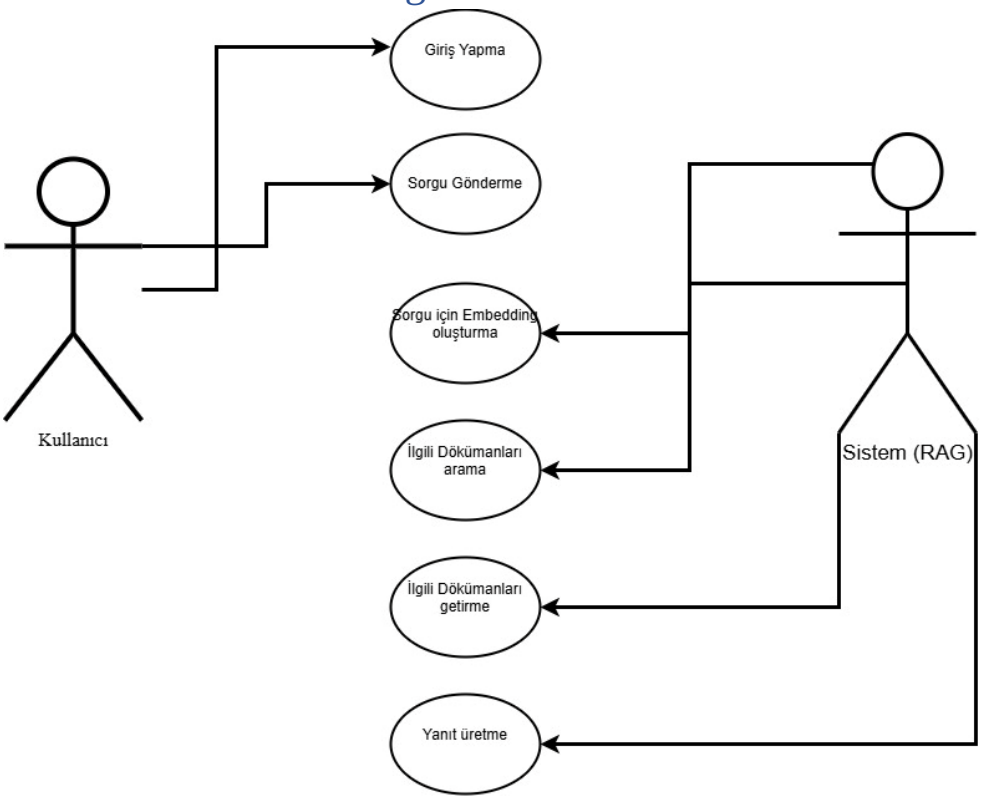
```python
def assign_category(row):
    similarities = {cat: cosine_similarity([row['aciklama_vector']], [vec]).item() for cat, vec in category_vectors.items()}
    return max(similarities, key=similarities.get)  #
```

2.4.4. LLM Sorgu Sonucunun Üretilmesi

- **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
```

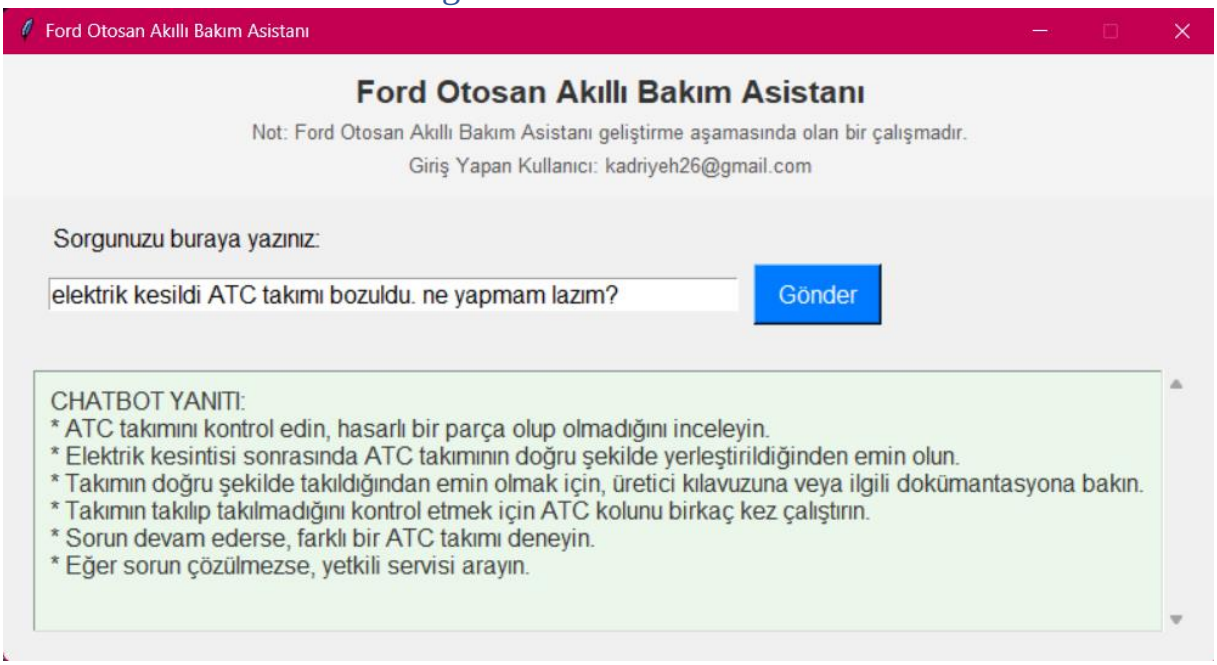
2.5. Use - Case Diagram



3. Fonksiyonel Olmayan Gereksinimler

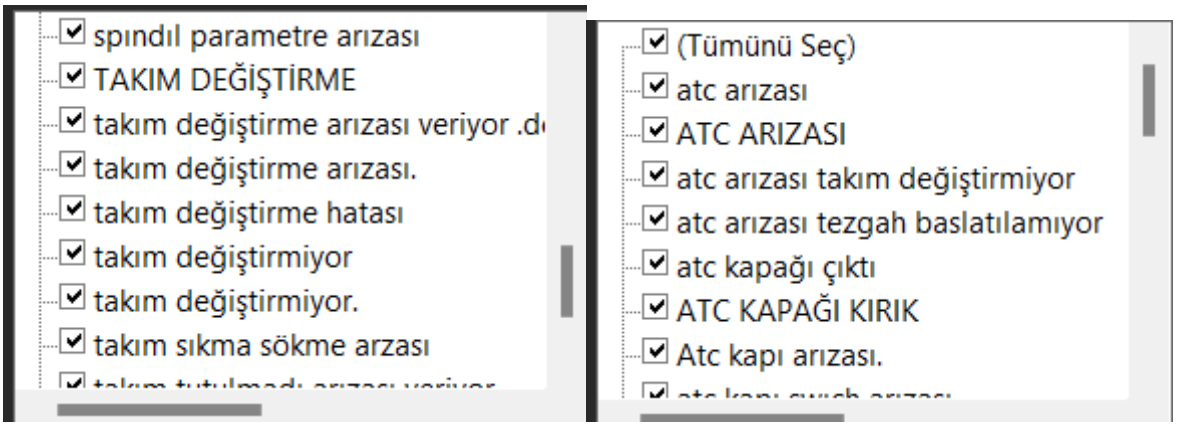
3.4. Kullanıcı Girdi-Sorgu Gereksinimleri

3.4.1. Örnek Girdi-Sorgu

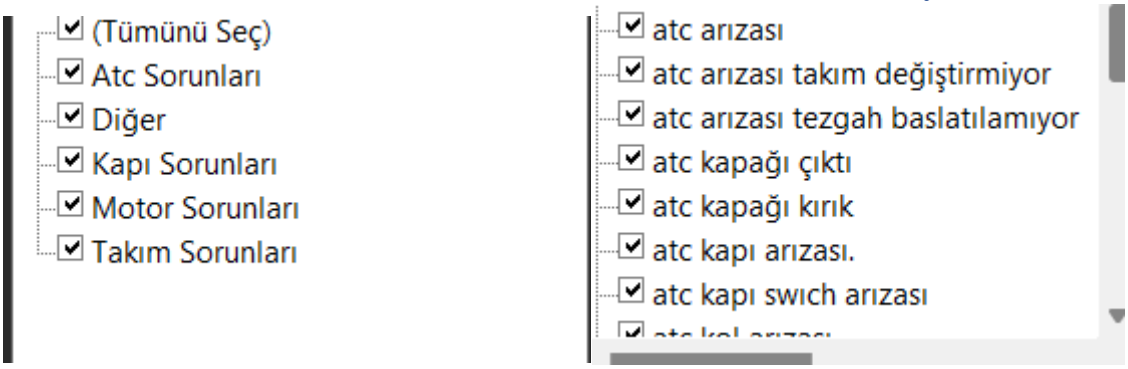


3.4.2. Kategorizasyon Hatalarının Eski ve Güncel Halleri

3.4.2.1. Eski Halleri



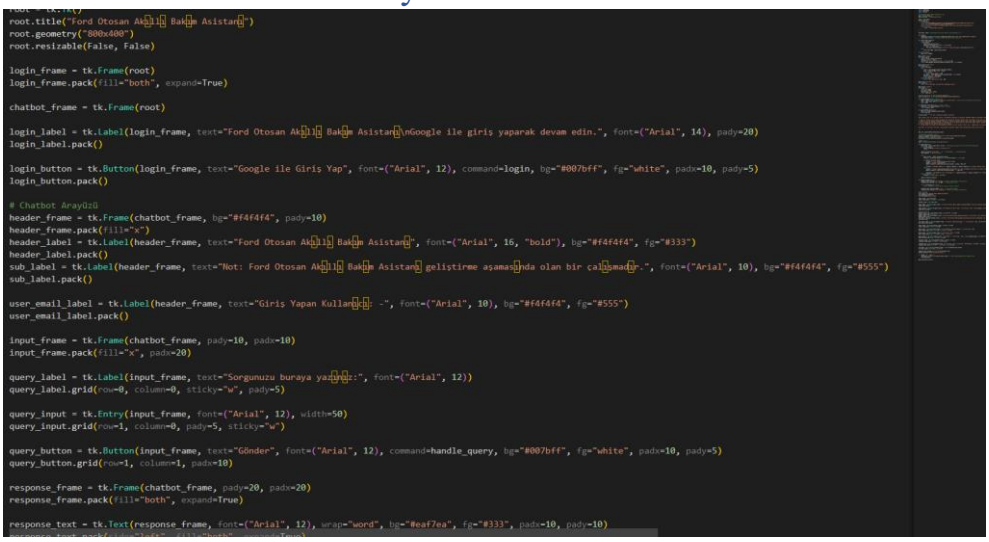
3.4.2.2. Veri Temizleme Sonrası Güncel Halleri ve Açıklama Sütunları



3.5. Arayüz Gereksinimleri

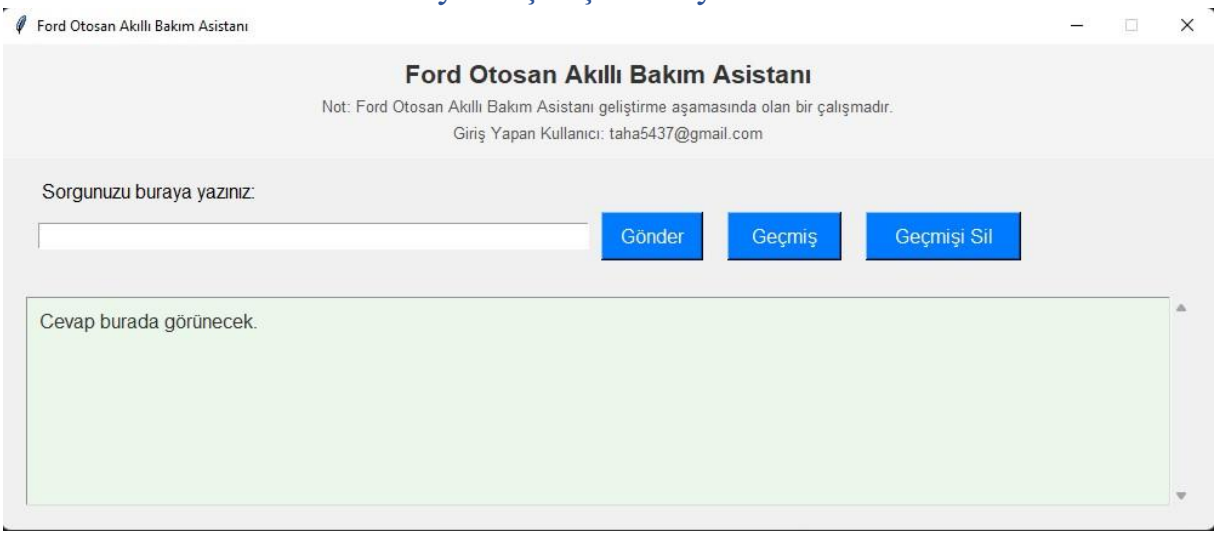
3.5.1. Taslak Kullanıcı Arayüzü

3.5.2. Kullanıcı Arayüz Kodları



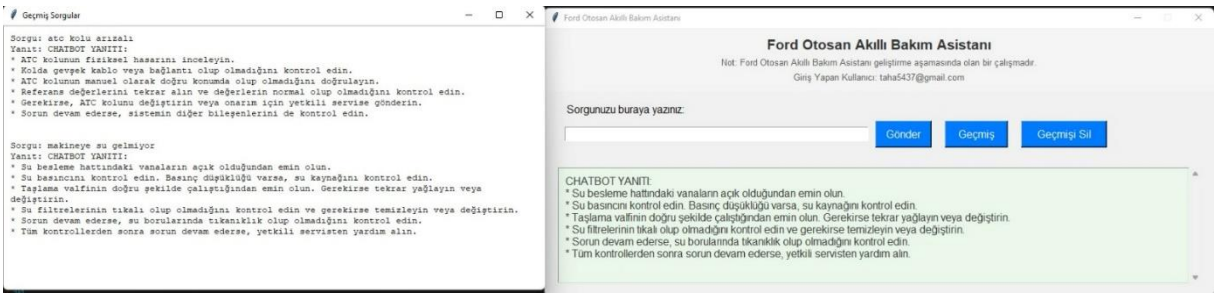
```
root.title("Ford Otosan Akıllı Bakım Asistanı")
root.geometry("800x600")
root.resizable(False, False)

login_frame = tk.Frame(root)
login_frame.pack(fill="both", expand=True)

chatbot_frame = tk.Frame(root)

login_label = tk.Label(login_frame, text="Ford Otosan Akıllı Bakım Asistanı\nGoogle ile giriş yaparak devam edin.", font=("Arial", 14), pady=20)
login_label.pack()

login_button = tk.Button(login_frame, text="Google ile Giriş Yap", font=("Arial", 12), command=login, bg="#007bff", fg="white", padx=10, pady=5)
login_button.pack()

# Chatbot Arayüzü
header_frame = tk.Frame(chatbot_frame, bg="#f4f4f4", pady=10)
header_frame.pack(fill="x")
header_label = tk.Label(header_frame, text="Ford Otosan Akıllı Bakım Asistanı", font=("Arial", 16, "bold"), bg="#f4f4f4", fg="#333")
header_label.pack()
sub_label = tk.Label(header_frame, text="Not: Ford Otosan Akıllı Bakım Asistanı geliştirme aşamasında olan bir çalışmadır.", font=("Arial", 10), bg="#f4f4f4", fg="#555")
sub_label.pack()

user_email_label = tk.Label(header_frame, text="Giriş Yapan Kullanıcı: -", font=("Arial", 10), bg="#f4f4f4", fg="#555")
user_email_label.pack()

input_frame = tk.Frame(chatbot_frame, pady=10, padx=10)
input_frame.pack(fill="x", padx=20)

query_label = tk.Label(input_frame, text="Sorgunuzu buraya yazınız:", font=("Arial", 12))
query_label.grid(row=0, column=0, sticky="w", pady=5)

query_input = tk.Entry(input_frame, font=("Arial", 12), width=50)
query_input.grid(row=1, column=0, pady=5, sticky="w")

query_button = tk.Button(input_frame, text="Gönder", font=("Arial", 12), command=handle_query, bg="#007bff", fg="white", padx=10, pady=5)
query_button.grid(row=1, column=1, padx=10)

response_frame = tk.Frame(chatbot_frame, pady=20, padx=20)
response_frame.pack(fill="both", expand=True)

response_text = tk.Text(response_frame, font=("Arial", 12), wrap="word", bg="#eaf7ea", fg="#333", padx=10, pady=10)
```

3.5.3. Planlanan Senaryoda Çalışan Arayüz Demosu



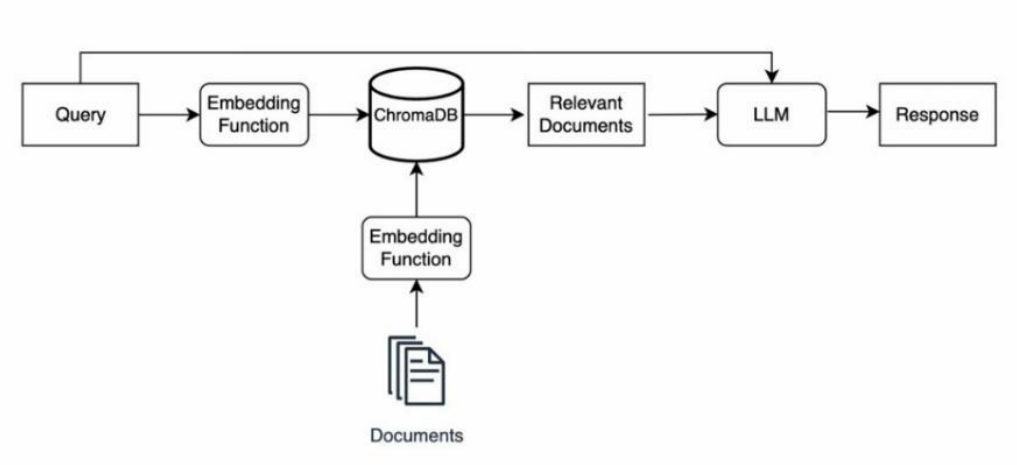

Ford Otosan Akıllı Bakım Asistanı
Ford Otosan Akıllı Bakım Asistanı
Not: Ford Otosan Akıllı Bakım Asistanı geliştirme aşamasında olan bir çalışmadır.
Giriş Yapan Kullanıcı: taha5437@gmail.com
Sorgunuzu buraya yazınız:
atc kolu arızalı
Gönder
Geçmiş
Geçmişi Sil
CHATBOT YANITI:
* ATC kolunun fiziksel hasarını 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.


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:
makineye su gelmiyor
Gönder
Geçmiş
Geçmişi 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.


Geçmiş Sorgular
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
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
Geçmişi 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.

3.6. Veri Tabanı Mimarisi ve Sistem Gereksinimleri

- RAG şeması şu şekildedir:



3.6.1. LLM API Konfigürasyonu ve Model Tanımlama

```
# Google Generative AI API anahtarını yapılandır
genai.configure(api_key="AIzaSyAOD73dXGPeN9d9rxlGNUTSCaPFR5GhPck")

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 yetkili
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 historic

Your goal is to deliver accurate, context-aware technical support while maintaining brevity and clarity.

"""

RAG_LLM = generateRAG_LLM(system_prompt)

# ChromaDB bağlantısını oluştur
client = chromadb.PersistentClient(path="C:/Users/Casper/Desktop/chromadb")
collection_name = "equipment_issues"
collection = client.get_collection(name=collection_name)

# Model Yükle
model = SentenceTransformer('all-MiniLM-L6-v2')
```

3.6.2. Exception Handling

```python
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."
        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.")
    except Exception as e:
        response = f"Hata oluştu: {str(e)}"

    update_response(response)

def update_response(text):
```

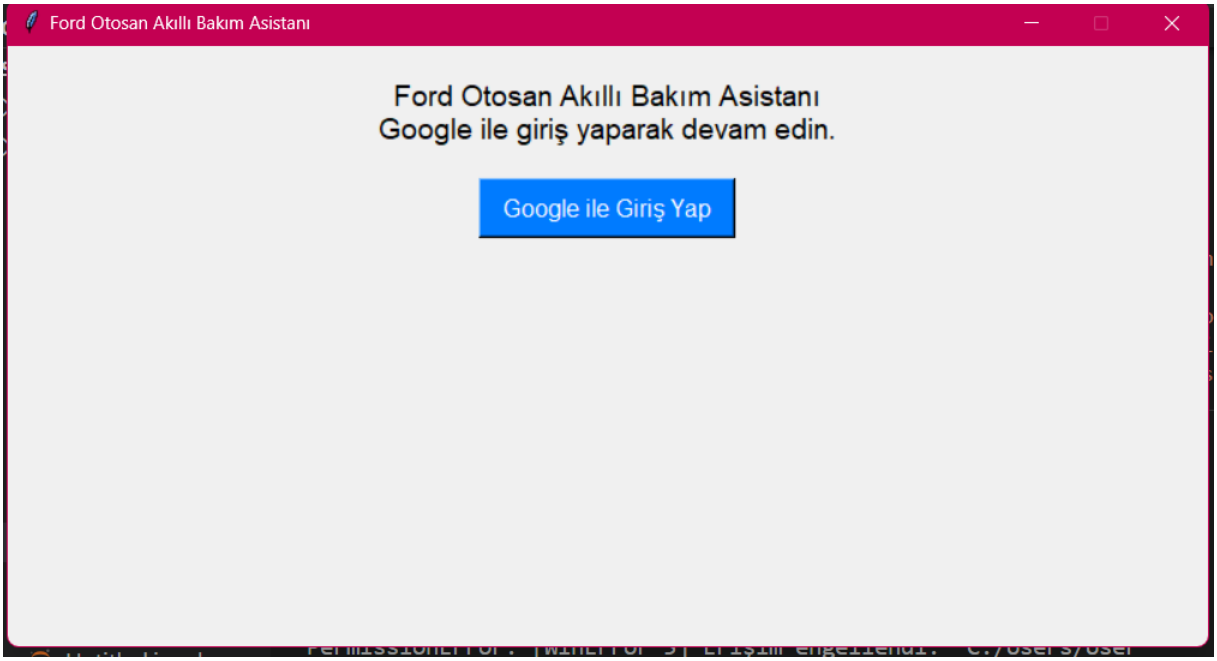
3.6.3. Veri Temizleme

```python
# Kategorileri tanımlayın
categories = {
    "Kapı Sorunları": ["kapı sıkışık", "kapı kapanmıyor", "kapı zor açılıyor"],
    "Motor Sorunları": ["motor çalışmıyor", "motor ses yapıyor", "motor sensör arızası"],
    "Elektrik Sorunları": ["elektrik kesintisi", "kablo kopuk"],
    "Takım Sorunları": ["takım değiştirme hatası", "takım sıkma sökme hatası"],
    "Atc Sorunları": ["atc arızası", "atc kol arızası", "atc arızası takım değiştirilemiyor", "atc arızası tezgah başlatılamıyor", "atc kapak arızası" , "atc kapı arızası",
    "Diğer": ["magazin hatası","robot arızası", "tezgah alarm", "tool clamper arızası", "pot takılacak", "shifter takılacak", "spindil takılacak"]
}

# Veri temizleme fonksiyonu: "arizasi"yi "arızası" olarak düzeltir ve büyük I harfini düşürürken ı olarak düzenler
def clean_text(text):
    text = text.replace("I", "ı")  # Büyük I harfini ı olarak düzelt
    text = re.sub(r'arizasi', 'arızası', text, flags=re.IGNORECASE)
    text = text.lower()  # Tüm harfleri küçük yap
    return text

# Veri yükleme
file_path = "veri.xlsx"
data = pd.read_excel(file_path)

# 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)

# Kategori vektörleri oluşturma
category_vectors = {category: model.encode(' '.join(phrases)) for category, phrases in categories.items()}

# Açıklamaları kategorilere ayırma
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)  # En yüksek benzerlik skoru olan kategoriyi döndür
data['Category'] = data.apply(assign_category, axis=1)

# Vektör sütunlarını kaldırma
data = data.drop(columns=['aciklama_vector'])
```

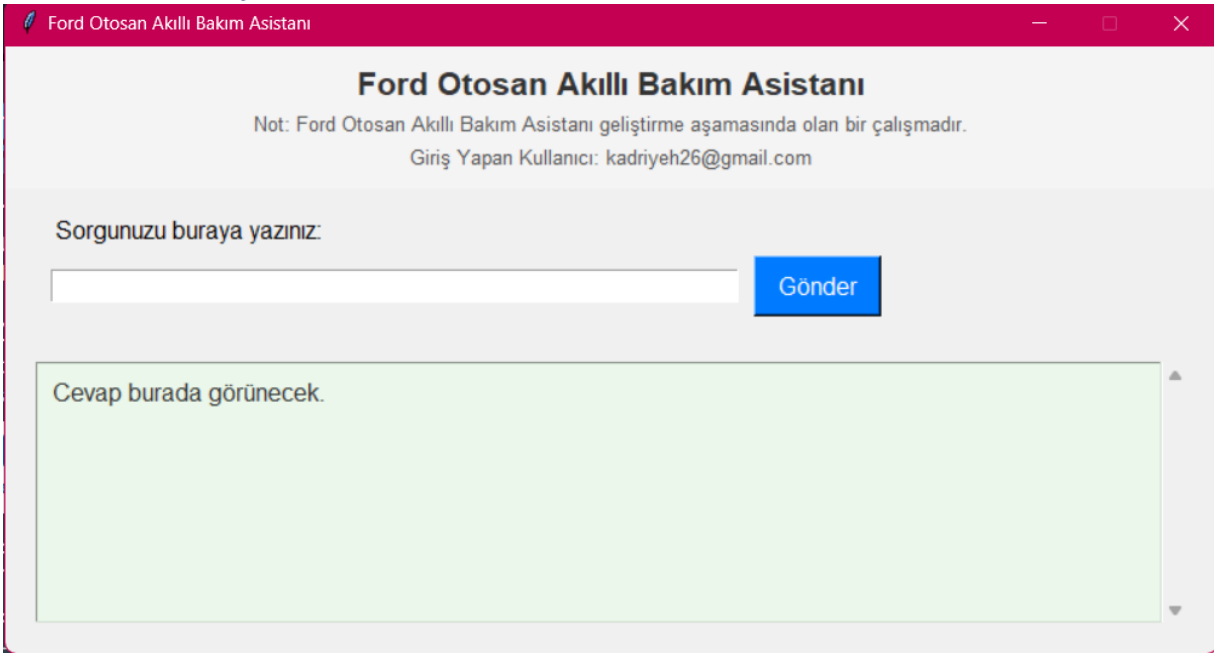
4. Diğer Entegrasyonlar

4.4. Google Hesabı ile Giriş Yapma

4.4.1. Authentication



4.4.2. Başarılı Authentication



4.4.3. Authentication Kod Kısmı

```python
def login():
    threading.Thread(target=lambda: webbrowser.open("http://127.0.0.1:5000/login")).start()
    check_login_status()  # Login durumunu düzenli olarak kontrol et

def check_login_status():
    global user_data
    if user_data:
        login_frame.pack_forget()
        chatbot_frame.pack(fill="both", expand=True)
        if 'email' in user_data:
            user_email_label.config(text=f"Giriş Yapan Kullan□□: {user_data['email']}")
    else:
        root.after(1000, check_login_status)

def start_flask():
    app.run(port=5000)

@app.route('/login')
def flask_login():
    nonce = secrets.token_urlsafe(16)
    session['nonce'] = nonce
    redirect_uri = url_for('authorize', _external=True)
    return oauth.google.authorize_redirect(redirect_uri, nonce=nonce)

@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

@app.route('/success')
def success():
    return "Giriş Başar□□□! Bu pencereyi kapatabilirsiniz."
```

4.4.4. XML

Geçmiş ve Geçmiş Sil özelliklerini sağlayan kod:

```python
# XML Kaydetme Fonksiyonu
def save_to_xml(user_email, query, response):
    filename = f"{user_email.replace('@', '_')}.xml"
    root = ET.Element("history") if not os.path.exists(filename) else ET.parse(filename).getroot()

    entry = ET.SubElement(root, "entry")
    ET.SubElement(entry, "query").text = query
    ET.SubElement(entry, "response").text = response

    tree = ET.ElementTree(root)
    tree.write(filename, encoding="utf-8", xml_declaration=True)

# XML Okuma Fonksiyonu
def read_xml(user_email):
    filename = f"{user_email.replace('@', '_')}.xml"
    if not os.path.exists(filename):
        return []

    root = ET.parse(filename).getroot()
    return [(entry.find("query").text, entry.find("response").text) for entry in root.findall("entry")]

# XML Temizleme Fonksiyonu
def clear_xml(user_email):
    filename = f"{user_email.replace('@', '_')}.xml"
    if os.path.exists(filename):
        os.remove(filename)


```

4.4.5. Import/Export (Excel):

```
# Veri yükleme
file_path = "veri.xlsx"
data = pd.read_excel(file_path)

# 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)

# Kategori vektörleri oluşturma
category_vectors = {category: model.encode(' '.join(phrases)) for category, phrases in categories.items()}

# Açıklamaları kategorilere ayırma
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)  # En yüksek benzerlik skoru olan kategoriyi döndür
data['Category'] = data.apply(assign_category, axis=1)

# Vektör sütunlarını kaldırma
data = data.drop(columns=['aciklama_vector'])

# Kategori sütununu Açıklama sütunundan sonra yerleştirme
cols = list(data.columns)
cols.insert(cols.index('Açıklama') + 1, cols.pop(cols.index('Category')))
data = data[cols]

# Kategorize edilmiş veriyi kaydetme
data.to_excel("categorized_veri.xlsx", index=False)
```

5. Proje Ekibi Değerlendirmesi

5.4. Çalışma-Adam-Saat Değerleri

İsim- Soy isim	**Projedeki Görevi**	**Adam - Saat**
Kadriye Harmancı	Veri Tabanı tasarımı, UI-Backend API araştırılması ve oluşturulması	64 saat
Abdullah Taha Aydın	VectorDB ve RAG sistemlerinin araştırılması ve oluşturulması	70 saat
Elif Suna Gegin	Veri Tabanı araştırması, Gereksinimlerin belirlenmesi, kontrolü ve hepsinin oluşturulması	60 saat
Umay Ece Mantar	Raporun düzenlenmesi, UI tasarımının araştırılması ve oluşturulması	60 saat
Toplam Süre		254 saat

6. Projenin Gelecek Durumu

Proje çalışmalarımız tamamlandığında, yapılan tüm geliştirme ve sonuçlar anlaşma gereği Ford Otosan'ın yazılım teknik birimine teslim edilecektir. Teslim sürecinde, projenin geliştirme

aşamaları, kullanılan teknolojiler ve gelinen son nokta detaylı bir şekilde teknik birime aktarılacaktır.

Ayrıca, sistem üzerinde yapılan veri temizleme, ayıklama ve yeniden kategorize etme işlemleri sonucunda elde edilen temizlenmiş ve yeniden kategorize edilmiş veri tabanı da teslim edilecektir. Bu veri tabanı, Ford Otosan'ın ilerideki ihtiyaçlarına yönelik olarak yeniden kullanılabilir ve geliştirilebilir bir yapıda olacaktır.

Teslim sırasında sistemin çalışmasını gösteren bir sunum veya demo yapılması da planlanmaktadır. Böylece, teknik birimin projeyi devralırken süreci eksiksiz anlaması ve projeyi ileriye taşıyabilmesi hedeflenmektedir.

7. Sonuç ve Genel Değerlendirme

7.4. Projenin Genel Değerlendirmesi

Bu proje sürecinde veri yönetimi konusunda ekip olarak önemli deneyimler kazanıldı. Alışık olunmayan bir yöntem olan ChromaDB ve uzun süredir ekip tarafından kullanılmayan Python ile çalışmak başta zorluk yaratsa da süreç içinde aşılmış ve öğrenim fırsatına dönüşmüştür.

UI kısmında müşteri beklentilerini karşılamanın önemi anlaşılırken, ekip içi iş birliği ve IEEE standartlarına uygun dokümantasyon hazırlama konularında pratik çözümler geliştirilmiştir.

Ekip üyeleri teknik ve kişisel becerilerini geliştirirken, veri yönetimi ve Excel gibi araçlar hakkında temel-orta seviye bilgi edinilmiş, teorik bilgilerin pratikte uygulanması sağlanmıştır.

7.5. Projenin Tam Entegrasyonunun Değerlendirmesi

Projenin tam entegrasyon aşamasında, geliştirilen tüm bileşenlerin uyumlu ve sorunsuz bir şekilde çalışması sağlanacaktır. **ChromaDB** ile veri tabanı entegrasyonu, **Google Gemini API** ile yanıt oluşturma süreçleri ve kullanıcı arayüzü (UI) arasındaki veri akışı test edilerek optimize edilmiştir.

Ayrıca, kimlik doğrulama işlemleri için **Google OAuth** entegrasyonu tam olarak devreye alınacak ve kullanıcı giriş süreçleri güvenli hale getirilmiştir. Sistemin **RAG (Retrieval-Augmented Generation)** mimarisi doğrultusunda çalıştığı doğrulanarak, kullanıcı sorgularının doğru şekilde işlenip yanıtlandığı kontrol edilmiştir.

Tam entegrasyon aşamasında sistemin farklı senaryolar altında performansı, hata yönetimi (Error Handling) ve veri tutarlılığı test edilecek; tüm bileşenlerin entegrasyonu sonunda sistem **kullanıcı beklentilerini** karşılayacak şekilde teslim edilecektir.

7.6. Projenin Alanda Kullanılabilirliği

Yaptığımız proje sayesinde, verilerin düzenlenmesi ve kategorize edilmesi şirketin bizden beklentisini karşılayarak, verilerin temizlenmesi ve işlenmesi sürecinde sağladığımız düzenleme, şirketin operasyonel verimliliğini artırmaya yönelik önemli bir adım olmuştur.

Proje, veri tabanının düzenlenmesi ve doğru şekilde kategorize edilmesi ile Ford Otosan'ın ihtiyaçlarına uygun hale getirilmiş, böylece projenin alanda kullanılabilirliği sağlanmıştır. Bu sayede, sistemin etkin bir şekilde kullanılabilmesi ve ileriye dönük veri yönetimi süreçlerinin iyileştirilmesi mümkün olmuştur.

7.7. Zaman Kalsaydı Yapılabilecekler

7.8. Sonuç

Öz değerlendirme Başlığı	**% xx (0-100)**
1-Proje başındaki amaç-hedeflerde değişme oranı	20
2- Mevcut veriyapısı (tasarım, oluşturulan tablo vb) nin proje ister ve uygulama ihtiyaçlarını karşılama oranı	100
3- Mevcut Karmaşık sorguların tasarım ve tam entegre uygulamaya alınma oranı;	100
4- Saklı Yordam, Hata Ayıklama **ve** Tetikleyicilerin tam entegre uygulamaya alınma oranı;	100
5- Arayüzlerin tamamlanması ve tam entegre uygulamaya alınma oranı	100
6- Diğer Entegrasyonların (sosyal medya, e-mail/gsm, export/import) tam entegre uygulamaya alınma oranı	90
7- Projenin tam entegre tamamlanma oranı	80
8-Uygulamanın alanda kullanılabilirlik oranı	90
9- Sunumun; genel akışı, teknik içerik aktarım yöntemleri, grup çalışanlarının kendi içinde dengeli şekilde konu anlatımı, zamanı etkin kullanım vb açısından değerlendirmesi	100
10- Proje raporunun proje çalışmalarını yansıtma oranı	100

8. Toplantılar

TARİH	**KATILIMCILAR**	**TOPLANTI KONUSU**	**TOPLANTI SONRASI ALINAN KARARLAR**
3.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Tanışma, proje konusu araştırması	Herkesin proje konusunu araştırmasına karar verildi.
9.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Proje için ortak olarak çalışabileceğimiz şirketlerin belirlenmesi, iletişim bilgilerinin listelenmesi	Firmalarla iletişime geçildi, görüşme ayarlanması için görev dağılımı yapıldı

13.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	FORD Otosan'da Product Manager Ömer Ersoy ALANYALI ile görüşme, projede ortaklığı kapsamında amacın ve işleyişin belirlenmesi ve gizlilik sözleşmesinin imzalanması	Konuya dair ilk araştırma için görev dağılımı yapıldı
18.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Şirketin teknik birim sorumlusu Senior Dev. Ahmet İCAT ile görüşme	Veri gereksinimleri hakkında bilgi edinildi ve başlangıç için örnek veriler teslim alındı
22.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	IEEE standartları, veri gereksinimleri hakkında araştırma ve notların sunumu	Gereksinim analizi raporu oluşturmak için görev dağılımı yapıldı
26.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN	Proje kapsamında kullanılacak teknolojiler belirlenmesi ve bilgi alışverişi	İlgili veri tabanı tasarımı, veri sorgularının hazırlanması ve verilen verinin kategorizasyonu
28.10.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Ara Rapor nihai formuna ulaştırma	Son rötuşların yapılması için görev dağılımı ve ilgili belgenin sisteme yüklenmesi
13.11.2024	Kadriye HARMANCI Abdullah Taha AYDIN	Güncel Gereksinimlerin Analizi	Rapor yazılması için görev dağılımı yapıldı.

	Elif Suna GEGİN Umay Ece MANTAR		
21.11.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Raporun gözden geçirilmesi	Sunum için görev dağılımı yapıldı.
29.11.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Alınan geri dönüşlere göre raporun yeniden düzenlenmesi için görev dağılımı	Raporda eksik bulunan yerler tartışıldı. Düzenleme için görev dağılımı yapıldı.
7.12.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Yapılan çalışmaların görüşülmesi	Frontend ve Backend kısmında yapılan çalışmalar paylaşıldı ve eksikler belirlendi.
15.12.2024	Kadriye HARMANCI Abdullah Taha AYDIN Elif Suna GEGİN Umay Ece MANTAR	Final rapor ve sunum hazırlığı	Rapor, sunum ve kodları nihai haline getirmek için görev dağılımı yapıldı.