Scrape Discord Bot

1. **Proje Özeti:**
   Projenin genel amacı, Discord botu aracılığıyla web sitelerinden veri kazımak, bu veriyi düzenlemek ve paylaşmaktır. Kullanıcılar botu kullanarak belirli web sitelerindeki verileri otomatik olarak alabilir ve e-posta ile paylaşabilirler. Bu, özellikle belirli bir alanda güncel bilgilere ihtiyaç duyan kullanıcılar için faydalı olabilir.

2. **Proje Gereksinimleri:**
   a. **Discord Bot Oluşturma:**

      Bir Discord botu oluşturulmalıdır. Bu bot, Discord sunucusunda komutlara yanıt verebilmelidir.

   b. **Web Sitelerinden Veri Kazıma:**

      Bot, belirli web sitelerinden veri kazıyabilmelidir. Bu web siteleri şunlar olmalıdır:
      - CBR Anime
      - Hashnode Data Science
      - Interesting Engineering
      - Wired Science
      - TechCrunch Startups

      Her web sitesi için ayrı bir kazıma işlemi ve işlevi olmalıdır.

   c. **Veri Toplama:**
      Her web sitesinden alınan veriler, bir veri çerçevesi (DataFrame) içinde birleştirilmelidir. Veri çerçevesi, aşağıdaki sütunları içermelidir:
      - "Site" (Web sitesi adı)
      - "Title" (Başlık)
      - "Link" (Bağlantı)
      - "Date" (Tarih)

      Veriler, bot tarafından bu veri çerçevesine eklenmelidir.

   ç. **Veri Kaydetme:**
      - Veri çerçevesi, pandas kütüphanesi kullanılarak bir Excel dosyasına kaydedilmelidir. Excel dosyasının adı "web_scraping_results.xlsx" olmalıdır.

d. E-posta Gönderme:

- Excel dosyasına kaydedilen veriler, bot tarafından e-posta ile belirli alıcılara gönderilmelidir.

- E-posta gönderme işlemi, SMTP protokolü kullanılarak gerçekleştirilmelidir. Aşağıdaki e-posta yapılandırması kullanılmalıdır:
  - SMTP sunucusu: smtp.gmail.com
  - SMTP port: 587
  - SMTP kullanıcı adı: e-mailadresi@gmail.com (Bu kısmı kendi e-posta adresinizle değiştirin)
  - SMTP parola: Oluşturulan uygulama parolanız (Gmail için güvenlik ayarlarından oluşturulabilir)
  - Alıcı e-posta adresleri, kod içinde belirtilmelidir. Birden fazla alıcı e-posta adresi desteklenmelidir.

- Alıcı e-posta adresleri, kod içinde belirtilmelidir. Birden fazla alıcı e-posta adresi desteklenmelidir.

e. Kontrollü Kapatma:

- Bot, belirli bir komutla kullanıcı tarafından kapatılabilir. Bu, botun aktif çalışmasını sonlandırır.

f. Hata İşleme:

- Proje, olası hataları ve sorunları ele almalıdır. Özellikle web sitelerine erişimde sorunlar veya e-posta gönderme sırasında hatalar gibi durumlar için hata işleme mekanizmaları eklenmelidir.

g. Kod Dökümantasyonu:

- Proje, anlaşılabilir ve düzenli bir kod yapısına sahip olmalıdır. Her bir işlev ve önemli kod bloğu açıklamalarla belgelenmelidir.

ğ. Kullanıcı Kılavuzu:

- Proje kullanıcılar için bir kılavuz içermelidir. Bu kılavuz, Discord sunucusuna botun nasıl eklenip kullanılacağını ve nasıl veri çıkarılacağını açıklamalıdır.

h. Güvenlik:

- Proje, kullanıcıların e-posta ve diğer hassas bilgilerini güvende tutmalıdır. SMTP parolaları gibi hassas bilgiler güvenli bir şekilde saklanmalıdır.

Bu gereksinimler, projenin temel işlevlerini ve beklenen davranışlarını açıklamaktadır. Projeyi başarıyla tamamlamak için bu gereksinimlere uygun bir şekilde çalışan bir Discord botu ve ilgili web sitelerinden veri çıkarma işlevleri geliştirilmelidir.

3. **Proje Tasarımı:**

   a. **Giriş:**
   Proje, belirli web sitelerinden veri kazıyan ve bu verileri bir Discord botu aracılığıyla kullanıcılarına e-posta ile gönderen bir uygulamayı hedefler.

   b. **Mimari:**
   - Proje, aşağıdaki ana bileşenleri içerir:
     - Discord Bot
     - Veri Kazıma Modülleri (Web Siteleri için)
     - Veri Toplama ve İşleme
     - E-posta Gönderme
     - Hata İşleme
     - Kullanıcı Arabirimi (Opsiyonel)

   - **Mimari Açıklamaları:**
     - Discord Bot, discord.py veya benzeri bir kütüphane kullanarak Discord sunucusunda komutlara yanıt verir.
     - Veri Kazıma Modülleri, her web sitesinden veri kazımını gerçekleştirir. Her bir web sitesi için ayrı bir modül olabilir.
     - Veri Toplama ve İşleme, verileri bir veri çerçevesinde birleştirir ve bir Excel dosyasına kaydeder.
     - E-posta Gönderme, SMTP protokolü kullanarak verileri alıcılara e-posta ile gönderir.
     - Hata İşleme, olası hataları ve sorunları ele alır ve hata durumlarında kullanıcıları bilgilendirir.
     - Kullanıcı Arabirimi (Opsiyonel), botun kullanımını kolaylaştıran bir arayüz sağlar (örneğin, Discord komutları).

   c. **Bot Etkinliği:**
     - **Bot, Discord sunucusunda belirli komutlara yanıt verir:**
       1. **!scrape: Web sitelerinden veri kazımını başlatır.**
       2. **!shutdown: Botu kapatır ve etkinliği sonlandırır.**

   ç. **Veri Kazıma Modülleri:**
     - Her bir web sitesi için ayrı modüller bulunur. Her modül, belirli bir web sitesindeki veriyi kazımak için gerekli kodları içerir.

d. **Veri Toplama ve İşleme:**
   - Veriler, her web sitesi kazıma modülünden alınır ve bir veri çerçevesi içinde birleştirilir.
   - Veriler, "Site", "Title", "Link" ve "Date" sütunlarında saklanır.

e. **E-posta Gönderme:**
   - Veriler, belirli alıcı e-posta adreslerine gönderilir.
   - E-posta yapısı, smtplib ve email modülleri kullanılarak oluşturulur.

f. **Hata İşleme:**
   - Proje, olası hataları ve sorunları ele alır ve kullanıcıları bilgilendirir. Özellikle web sitelerine erişimde sorunlar veya e-posta gönderme sırasında hatalar için hata işleme mekanizmaları eklenir.

g. **Kullanıcı Kılavuzu:**
   - Kullanıcılar için bir kılavuz sağlanır. Bu kılavuz, Discord sunucusuna botun nasıl eklenip kullanılacağını ve veri kazıma işleminin nasıl başlatılacağını açıklar.

ğ. **Güvenlik:**
Hassas bilgiler (örneğin, SMTP parolaları), güvenli bir şekilde saklanır ve korunur.

4. **Proje Uygulaması:**
Bu bölümde projemizin kodlarının kısa bir açıklaması ve çalışmanın ekran görüntüleri paylaşılmıştır.

- Gerekli kütüphanelerin import edilmesi:

```
import discord
from discord.ext import commands
import requests
from bs4 import BeautifulSoup
import pandas as pd
import smtplib
from urllib.parse import urljoin
from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText
from email.mime.base import MIMEBase
from email.mime.application import MIMEApplication
from email import encoders
import nest_asyncio
```

- Bot intents işlemleri ve ayarlamalar:

```
nest_asyncio.apply()
#`nest_asyncio.apply()`, asyncio'yu bir Jupyter not defterinde veya
halihazırda çalışan bir olay döngüsüne sahip bir ortamda
kullanmanızı sağlayan bir fonksiyondur.
# İç içe geçmiş asyncio olay döngülerinin düzgün çalışmasını
sağlamak için olay döngüsüne yama yapar.

intents= discord.Intents.all()
intents.members=True
intents.message_content = True
# Kod, `discord` modülünden `Intents` sınıfının bir örneğini
oluşturuyor ve bunu `intents` değişkenine atıyor.

bot = commands.Bot(command_prefix='!', intents=intents)
""" `commands` modülünden `Bot` sınıfının bir örneğini oluşturur.
Komut önekini `'!'` olarak ve niyetleri `intents` değişkenine
ayarlar.
Bu kod satırı tipik olarak bir Discord botu için bir bot nesnesi
oluşturmak için kullanılır;
burada komut öneki botun komutlara nasıl yanıt vereceğini ve
amaçlar botun hangi olayları dinleyebileceğini belirler.
"""
```

- Botun başlangıçta isminin yazılması işlemleri

```
@bot.event
async def on_ready():
    print(f'Logged in as {bot.user.name}')

# on_ready` işlevi, oturum açıldığında botun adını yazdıran bir
olay işleyicisidir.
```

“ !Discord botunun oluşturulma ve aktif edilme işlemleri Discord_bot_creating.ipynb isimli kod dosyasında ve aynı isimli klasördeki bot oluşturma bölümünde anlatılmıştır.”

- Bot komut işlemleri:

```
@bot.command(name='scrape')
async def scrape(ctx):
  """
scrape` fonksiyonu botun bir web sitesinden veri
kazımasını sağlayan bir komuttur.

Args
  ctx: ctx` parametresi "bağlam" anlamına gelir ve komutun
yürütüldüğü bağlamı temsil eder. Mesaj, mesajı gönderen
kullanıcı, mesajın gönderildiği kanal ve diğer ilgili
ayrıntılar hakkında bilgi içerir.
  """
```

- Veri kazıma ve listeye kaydetme işlemleri

```
def scrape_and_save_data():
        data = {
            'Site': [],
            'Title': [],
            'Link': [],
            'Date': []
        }

        def scrape_cbr_anime():
            site_name = "CBR/Anime"
            base_url = 'https://www.cbr.com/'

            url = 'https://www.cbr.com/category/anime/'

            response = requests.get(url)

            if response.status_code == 200:
                soup = BeautifulSoup(response.text,
'html.parser')
                article_blocks = soup.find_all('div',
class_='w-display-card-content')
                for block in article_blocks:
                    title_element = block.find('h5',
class_='display-card-title').find('a')
                    title = title_element.text.strip()
                    relative_link = title_element['href']
```

```
                    full_link = urljoin(base_url,
relative_link)
                    date_element = block.find('time',
class_='display-card-date')['datetime']
                    date = date_element.split('T')[0]

                    data['Site'].append(site_name)
                    data['Title'].append(title)
                    data['Link'].append(full_link)
                    data['Date'].append(date)

                # Burdaki kodların temel işlevi
CBR/Anime web sitesindeki verileri kazır ve bir sözlüğe
kaydeder.
```

“ Her bir site için benzer işlemler gerçekleştirilmiştir. Hangi siteden veri çekmek istiyorsanız sitenin HTML kodlarını dikkatli bir biçimde incelemeniz ve etiketleri buna göre düzenlemeniz gerekmektedir.”

- Her bir site için kazınan verileri çağırma ve dataframe çıkarma ve kaydetme:

```
# Her site için kazıma işlevlerini çağırın
        scrape_cbr_anime()
        scrape_hashnode_data_science()
        scrape_interesting_engineering()
        scrape_wired_science()
        scrape_techcrunch_startups()
        # Toplanan verilerden bir DataFrame oluşturun
        df = pd.DataFrame(data)

        """
        Bu işlevler CBR Anime, Hashnode Data Science, Interesting
Engineering, Wired Science ve TechCrunch Startups gibi web
sitelerinden veri kazıyor.
        Verileri kazıdıktan sonra, toplanan verilerden bir
DataFrame oluşturur ve bunu "web_scraping_results.xlsx" adlı bir
Excel dosyasına kaydeder.
        """

# DataFrame'i bir Excel dosyasına kaydedin
        df.to_excel('web_scraping_results.xlsx', index=False)
```

- E-posta yapılandırması:

```
# E-posta yapılandırması
        smtp_server = 'smtp.gmail.com'
        smtp_port = 587
        smtp_username = 'gönderenmail@gmail.com'  # Gmail
e-posta adresinizle değiştirin
        smtp_password = 'uygulama_parolası'  # Oluşturulan
uygulama parolanızla değiştirin

        """
        e-posta göndermek için SMTP sunucusu, bağlantı
noktası, kullanıcı adı ve parola ile e-posta
yapılandırmasını ayarlar.
        """

        # Alıcı e-posta adresleri
        recipient_emails = ['alıcı1@outlook.com',
'alıcı2@gmail.com']  # Alıcı e-posta adreslerinizle
değiştirin
        """
        Bu e-posta adreslerinin bir e-postanın alıcıları
olması amaçlanmıştır. Açıklama `# Alıcı e-posta adresleri`
sadece kodun amacını belirtmek için açıklayıcı bir
yorumdur.
        Çoklu e-posta adresleri ekleyerek birden fazla
mail adresine gönderim yapılmaktadır.
        """

        # Email içeriği
        body = 'Please find the attached web scraping
results.'
        # e-posta gövdesinin içeriğini saklamak için
kullanılan `body` değişkenine atar.

        # Excel dosyasını ekleyin
        with open('web_scraping_results.xlsx', 'rb') as
```

```
file:
            attachment = MIMEApplication(file.read(),
_subtype="xlsx")
            attachment.add_header('Content-Disposition',
'attachment', filename='web_scraping_results.xlsx')

            """
            Kod parçacığı `open()` fonksiyonunu kullanarak
ikili modda 'web_scraping_results.xlsx' adlı bir dosya
açmaktadır.
            Daha sonra `read()` yöntemini kullanarak
dosyanın içeriğini okur ve `attachment` değişkenine atar.
Burada attachment = ek dosyası olarak atanan scraping
işlemlerinin kaydedildiği excel dosyasıdır.
```

- E-posta gönderme işlemleri:

```
# E-postayı her alıcıya ayrı ayrı gönderin
        for recipient_email in recipient_emails:
            msg = MIMEMultipart()
            msg['From'] = smtp_username
            msg['To'] = recipient_email
            msg['Subject'] = 'Web Scraping Results'

            msg.attach(MIMEText(body, 'plain'))
            msg.attach(attachment)

            server = smtplib.SMTP(smtp_server, smtp_port)
            server.starttls()
            server.login(smtp_username, smtp_password)
            server.sendmail(smtp_username,
recipient_email, msg.as_string())
            server.quit()

        print('Email sent successfully to the
recipients:', ', '.join(recipient_emails))
        """Kod, SMTP protokolünü kullanarak birden fazla
alıcıya e-posta gönderiyor. Dosya gönderildikten sonra
Email başarı ile gönderildi mesajı gelir ve mail'in
gönderildiği adresler ekrana
        çıktı olarak yazılır."""
```

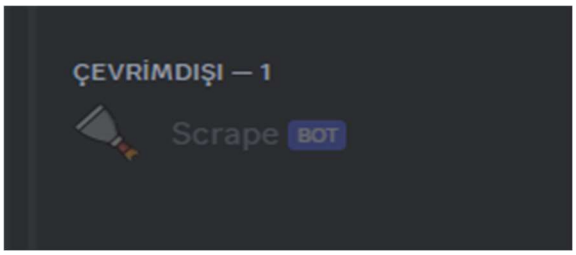
- Botu manuel olarak durdurma işlemleri:

```
@bot.command(name='shutdown')
async def shutdown(ctx):
    await ctx.send('Shutting down...')
    await bot.close()

    """
    Shutdown` fonksiyonu, botun kapatıldığını belirten bir mesaj gönderen ve ardından botu kapatan bir komuttur.

    Argümanlar
      ctx: ctx` parametresi "bağlam" anlamına gelir ve komutun çağrıldığı bağlamı temsil eder. Mesaj, mesajı gönderen kullanıcı, mesajın gönderildiği kanal vb. hakkında bilgi içerir.
    """
```

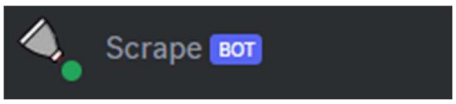
- Botun çalıştırılması:

```
bot.run('your_bot_token') # Discord botunun çalışması için run fonksiyonu kullanılır. Burada tırnak içerisine kendi bot token kodunuzu yapıştırmanız gerekir.
```

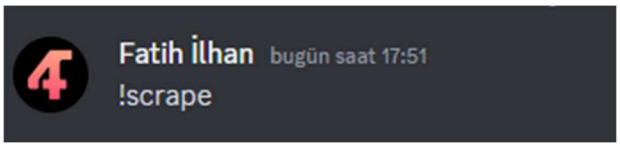
5. **Proje Testi:**
   - Botun sunucudaki çevrimdışı görüntüsü:



Botun sunucuda kod bloğunu çalıştırınca çevrimiçi hale gelmesi:

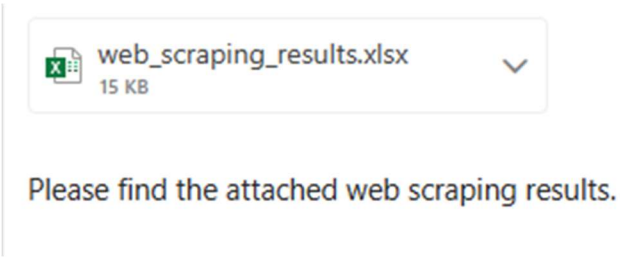


- Discord komutunun yazılması:

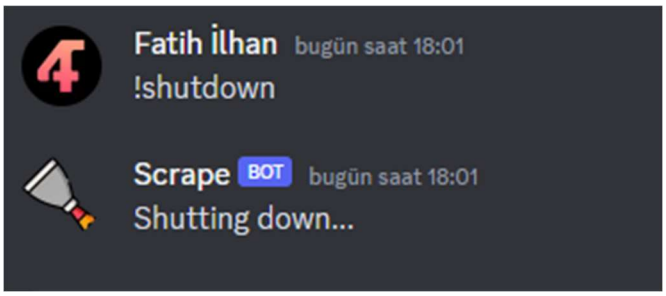


- Kazıma kodunun çalışması ve verilerin çıkarılıp, kaydedilip mail gönderim işlemlerinin başarıyla tamamlanması:

- E-mail mesaj metni ve içeriği:



- Botun manuel olarak komutla kapatılması:



- Ve çıkarılan verilerin kontrol edilmesi:

Site	Title	Link	Date
CBR/Anime	From Casca to Fena: Anime's Fascinati	https://www.cbr.com/joan-of-arc	2023-09-27
CBR/Anime	10 Forgotten 2000s Anime You Watche	https://www.cbr.com/forgotten-2	2023-09-27
CBR/Anime	How Dragon Quest Unintentionally Cre	https://www.cbr.com/anime-trop	2023-09-27
CBR/Anime	10 Important Anime Moments From th	https://www.cbr.com/iconic-anim	2023-09-27
CBR/Anime	10 Best Horror Tropes In Berserk, Rank	https://www.cbr.com/berserk-ho	2023-09-26
CBR/Anime	Girl's Love Anime I'm In Love With The	https://www.cbr.com/girls-love-ir	2023-09-26
CBR/Anime	Spy X Family Season 2 Opening and En	https://www.cbr.com/spy-x-famil	2023-09-26
CBR/Anime	Top 10 Most Anticipated New Anime o	https://www.cbr.com/hyped-my-	2023-09-26
CBR/Anime	Mobile Suit Gundam Seed Freedom Fil	https://www.cbr.com/mobile-suit	2023-09-26
CBR/Anime	10 Biggest Differences Between DBZ's	https://www.cbr.com/dbzs-vs-dra	2023-09-26
CBR/Anime	Scott Pilgrim Takes Off Will Feature "U	https://www.cbr.com/scott-pilgri	2023-09-26
CBR/Anime	15 Must-Watch Fantasy Anime With Th	https://www.cbr.com/best-romar	2023-09-26
CBR/Anime	JJK: How Yuji Itadori Can Improve His C	https://www.cbr.com/jjk-yuji-itad	2023-09-26
CBR/Anime	Cozy Crafting: Best Manga About Knitti	https://www.cbr.com/best-cozy-c	2023-09-26
CBR/Anime	10 Best Fantasy Tropes in Bleach, Rank	https://www.cbr.com/bleach-fant	2023-09-26
CBR/Anime	The 20 Most Disliked Demon Slayer Ch	https://www.cbr.com/demon-sla	2023-09-26
CBR/Anime	Best Harem Anime, Ranked	https://www.cbr.com/best-harem	2023-09-26
CBR/Anime	10 Best Manga Classics, Ranked	https://www.cbr.com/best-classic	2023-09-26
CBR/Anime	One Piece: The Final Straw Hat Was Re	https://www.cbr.com/one-piece-	2023-09-26
CBR/Anime	10 Moments That Made Fans Fall in Lo	https://www.cbr.com/melancholy	2023-09-26
Hashnode/Data Science	The Only Roadmap to Data Science : Y	https://thedataalchemist.hashno	Sep 25, 2023

Sonuç:

Bu proje, veri kazıma ve iletişim işlemlerini bir araya getiren etkileyici bir Discord botunu hayata geçirmek için birlikte çalıştığımız harika bir deneyimdi. Bu projenin başarılı bir şekilde tamamlanmasına katkıda bulunan ve projenin her aşamasında önemli bir rol oynayan dostum Fatih İlhan'a teşekkürlerimi sunmak isterim. İşbirliğimiz sayesinde, projeyi başarılı bir şekilde geliştirdik ve sonuçlarını görmek büyük bir memnuniyet kaynağı oldu.

Projemizin temel amacı, belirli web sitelerinden veri kazıma işlemini otomatize etmek ve bu verileri Discord botu aracılığıyla kullanıcılarımıza sunmaktı. Bu amaç doğrultusunda, farklı web siteleri için veri kazıma modülleri geliştirdik ve bu verileri düzenli bir şekilde saklayıp işledik. Ayrıca, kazıdığımız verileri e-posta yoluyla kullanıcılara ulaştırmak için SMTP protokolünü kullandık.

Bu projenin geliştirilmesi sırasında aşağıdaki anahtar noktaları başarıyla ele aldık:

- Python programlama dili kullanarak Discord botunu oluşturma.
- Çeşitli web sitelerinden veri kazıma işlemini otomatikleştirme.
- Kazıdığımız verileri bir veri çerçevesinde düzenleme ve saklama.
- SMTP protokolünü kullanarak e-posta gönderme işlemi.
- Projenin düzgün çalışması ve güvenliği için gerekli önlemleri alma.
- Projemizin sonuçları, veri kazıma işleminin başarıyla gerçekleştirdiğini ve kazıdığımız verilerin kullanıcılarımıza başarılı bir şekilde iletilmekte olduğunu göstermektedir. Bu projenin bize öğrettiği önemli bir ders, işbirliği ve problem çözme yeteneklerinin teknoloji projelerini nasıl daha etkili hale getirebileceği

dir.

Sonuç olarak, bu projenin başarıyla tamamlanması, harika bir ekip çalışmasının ve kararlılığın bir sonucudur. Fatih İlhan'a bu projede gösterdiği katkılar için teşekkür ederim ve gelecekteki işbirliklerimizi dört gözle bekliyorum.

Saygılarımızla,

Fatih İlhan Yusuf Çınarcı