**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.

1. **Proje Gereksinimleri:**
   1. **Discord Bot Oluşturma:**

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

* 1. **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.

* 1. **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.

* 1. **Veri Kaydetme:**
* Veri çerçevesi, pandas kütüphanesi kullanılarak bir Excel dosyasına kaydedilmelidir. Excel dosyasının adı "web\_scraping\_results.xlsx" olmalıdır.
  1. **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.
  1. **Kontrollü Kapatma:**
* Bot, belirli bir komutla kullanıcı tarafından kapatılabilir. Bu, botun aktif çalışmasını sonlandırır.
  1. **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.
  1. **Kod Dökümantasyonu:**
* Proje, anlaşılabilir ve düzenli bir kod yapısına sahip olmalıdır. Her bir işlev ve önemli kod bloğu açıklamalarla belgelenmelidir.
  1. **Kullanıcı Kılavuzu:**
* Proje kullanıcılar için bir kılavuz içermelidir. Bu kılavuz, Discord sunucusuna botun nasıl eklenip kullanılacağını ve nasıl veri çıkarılacağını açıklamalıdır.
  1. **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.

1. **Proje Tasarımı:**
2. **Giriş:**

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

1. **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).

1. **Bot Etkinliği:**
   * **Bot, Discord sunucusunda belirli komutlara yanıt verir:**
     1. **!scrape: Web sitelerinden veri kazımını başlatır.**
     2. **!shutdown: Botu kapatır ve etkinliği sonlandırır.**
2. **Veri Kazıma Modülleri:**
   * Her bir web sitesi için ayrı modüller bulunur. Her modül, belirli bir web sitesindeki veriyi kazımak için gerekli kodları içerir.
3. **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.
4. **E-posta Gönderme:**
   * Veriler, belirli alıcı e-posta adreslerine gönderilir.
   * E-posta yapısı, smtplib ve email modülleri kullanılarak oluşturulur.
5. **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.
6. **Kullanıcı Kılavuzu:**
   * Kullanıcılar için bir kılavuz sağlanır. Bu kılavuz, Discord sunucusuna botun nasıl eklenip kullanılacağını ve veri kazıma işleminin nasıl başlatılacağını açıklar.
7. **Güvenlik:**

Hassas bilgiler (örneğin, SMTP parolaları), güvenli bir şekilde saklanır ve korunur.

1. **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:

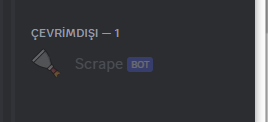
|  |
| --- |
| @bot.command(name='shutdown') async def shutdown(ctx):  await ctx.send('Shutting down...')  await bot.close()   """  Shutdown` fonksiyonu, botun kapatıldığını belirten bir mesaj gönderen ve ardından botu kapatan bir komuttur.   Argümanlar  ctx: ctx` parametresi "bağlam" anlamına gelir 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. |

1. **Proje Testi:**

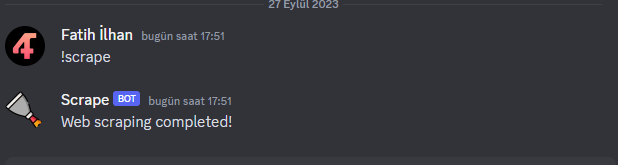
* Botun sunucudaki çevrimdışı görüntüsü:



Botun sunucuda kod bloğunu çalıştırınca çevrimiçi hale gelmesi:



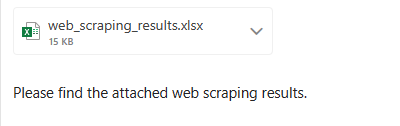
* Discord komutunun yazılması:



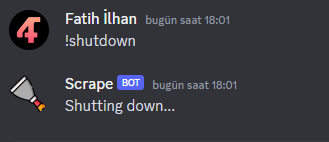
* Kazıma kodunun çalışması ve verilerin çıkarılıp, kaydedilip mail gönderim işlemlerinin başarıyla tamamlanması:



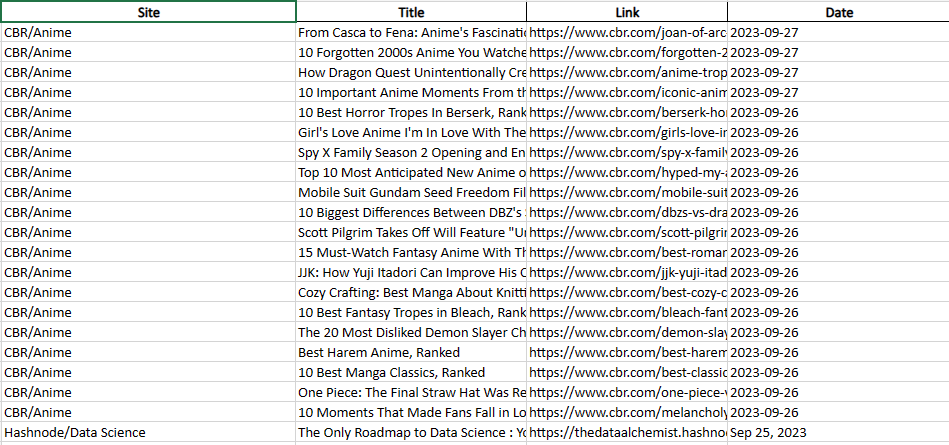
* E-mail mesaj metni ve içeriği:



* Botun manuel olarak komutla kapatılması:



* Ve çıkarılan verilerin kontrol edilmesi:



**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