**scraper/**

**1. \_\_init\_\_.py**

* **Amacı**: Bu dosya, scraper modülünün bir paket olarak tanınmasını sağlar.
* **İçerik**: Genellikle boş bırakılır, ancak başka modüllerden içe aktarım için kullanılabilir.
* **Fonksiyonlar**: Genellikle fonksiyon içermez. Ancak gerekli durumlarda başlangıçta yapılması gereken ayarlar veya içe aktarımlar yapılabilir.

**2. main.py**

* **Amacı**: Projenin başlangıç noktasıdır ve kullanıcıdan veri alarak kazıma işlemlerini başlatır.
* **İçerik**:
  + Komut satırından URL alma.
  + config.py dosyasından ayarları okuma.
  + Kazıma işlemini başlatma.
  + Geri bildirim için loglama.
* **Fonksiyonlar**:
  + def main(): - Uygulamanın ana akışını yöneten fonksiyon.
  + def get\_user\_input(): - Kullanıcıdan URL ve diğer girişleri almak için.
  + def load\_config(): - Konfigürasyon ayarlarını yükler.

**3. config.py**

* **Amacı**: Tüm konfigürasyon ayarlarını saklar, böylece değişiklik yapmak kolaydır.
* **İçerik**:
  + Proxy ayarları.
  + Kullanıcı ajanları listesi.
  + Diğer ayar parametreleri.
* **Fonksiyonlar**:
  + def load\_config(): - Konfigürasyon ayarlarını yükler ve bir Python sözlüğü olarak döner.
  + def save\_config(config): - Güncellenmiş konfigürasyonu dosyaya kaydeder.

**4. scraper.py**

* **Amacı**: E-ticaret sitelerinden veri çekmek için ana mantığı içerir.
* **İçerik**:
  + HTTP istekleri ve yanıt işleme.
  + Çekilen verilerin analiz edilmesi.
* **Fonksiyonlar**:
  + class Scraper: - Kazıma işlemleri için ana sınıf.
    - def \_\_init\_\_(self, url): - URL ile nesneyi başlatır.
    - def fetch\_data(self): - Belirtilen URL'den veri çeker.
    - def parse\_response(self, response): - Yanıtı işler.
    - def extract\_product\_info(self, html): - Ürün bilgilerini çıkarır.

**utils/**

**1. request\_utils.py**

* **Amacı**: HTTP istekleri için yardımcı fonksiyonlar sağlar.
* **İçerik**:
  + HTTP istekleri gönderme ve yanıtları işleme.
* **Fonksiyonlar**:
  + def send\_request(url, method='GET', headers=None, params=None): - HTTP isteği gönderir ve yanıtı döner.
  + def handle\_response(response): - Yanıtı işler ve hata durumlarını yönetir.

**2. parse\_utils.py**

* **Amacı**: HTML içeriğinden veri çıkarmaya yardımcı olan fonksiyonları içerir.
* **İçerik**:
  + Verileri temizleme ve formatlama.
* **Fonksiyonlar**:
  + def clean\_data(data): - Ham veriyi temizler ve normalize eder.
  + def extract\_data(html\_content, selectors): - Belirtilen CSS seçiciler ile veriyi çıkarır.

**3. wait\_utils.py**

* **Amacı**: Kazıma sırasında insan benzeri bekleme süreleri ekler.
* **İçerik**:
  + Gecikme sürelerini yöneten fonksiyonlar.
* **Fonksiyonlar**:
  + def sleep\_random(min\_time, max\_time): - Rastgele bir süre bekler.

**4. user\_agent\_utils.py**

* **Amacı**: Kullanıcı ajanlarını dinamik olarak yönetir.
* **İçerik**:
  + Kullanıcı ajanı listesini yükler ve seçer.
* **Fonksiyonlar**:
  + def load\_user\_agents(): - Kullanıcı ajanları listesini yükler.
  + def get\_random\_user\_agent(): - Rastgele bir kullanıcı ajanı döner.

**5. proxy\_utils.py**

* **Amacı**: Proxy ayarlarını yönetir.
* **İçerik**:
  + Proxy listesini yükler ve doğrular.
* **Fonksiyonlar**:
  + def load\_proxies(source): - Proxy listesini belirli bir kaynaktan yükler.
  + def get\_random\_proxy(): - Rastgele bir proxy döner.
  + def validate\_proxy(proxy): - Proxy'nin geçerliliğini kontrol eder.

**site\_detection/**

**1. site\_detector.py**

* **Amacı**: E-ticaret sitelerinin HTML yapılarını analiz eder.
* **İçerik**:
  + HTML yapılarını tanıyan algoritmalar.
* **Fonksiyonlar**:
  + def detect\_site\_structure(html\_content): - HTML içeriğini analiz eder ve yapı hakkında bilgi verir.

**2. site\_config.py**

* **Amacı**: Belirli sitelere özgü konfigürasyonları yönetir.
* **İçerik**:
  + Siteye özgü ayarları içerir.
* **Fonksiyonlar**:
  + def get\_site\_specific\_config(site): - Belirli bir site için konfigürasyonu döner.

**data\_parsers/**

**1. data\_parser.py**

* **Amacı**: Kazıma işlemi sonucu elde edilen verileri işler.
* **İçerik**:
  + Verileri kaydetme ve analiz etme.
* **Fonksiyonlar**:
  + def save\_data(data, format='csv'): - Veriyi belirli bir formatta kaydeder.
  + def process\_data(raw\_data): - Ham veriyi işler.

**2. html\_parser.py**

* **Amacı**: HTML yanıtlarını işlemek için alternatif yöntemler sunar.
* **İçerik**:
  + HTML analizini gerçekleştiren kütüphaneler.
* **Fonksiyonlar**:
  + def parse\_html(html): - HTML içeriğini analiz eder ve yapısını döner.

**proxy\_manager/**

**1. proxy\_manager.py**

* **Amacı**: Proxy havuzunun yönetimini sağlar.
* **İçerik**:
  + Proxy geçerliliğini kontrol etme.
* **Fonksiyonlar**:
  + def manage\_proxies(): - Proxy havuzunu yönetir.
  + def remove\_invalid\_proxies(): - Geçersiz proxy’leri havuzdan çıkarır.

**2. proxy\_loader.py**

* **Amacı**: Proxy kaynaklarından proxy bilgilerini yükler.
* **İçerik**:
  + Proxy yükleme işlemleri.
* **Fonksiyonlar**:
  + def load\_proxies\_from\_file(file\_path): - Dosyadan proxy yükler.
  + def load\_proxies\_from\_api(api\_url): - API’den proxy yükler.

**3. proxy\_rotator.py**

* **Amacı**: Kazıma sırasında proxy değişimini otomatikleştirir.
* **İçerik**:
  + Proxy değişim mantığı.
* **Fonksiyonlar**:
  + def rotate\_proxy(): - Proxy rotasyonunu gerçekleştirir.

**4. proxy\_validator.py**

* **Amacı**: Proxy’lerin geçerliliğini kontrol eder.
* **İçerik**:
  + Geçerli proxy’leri tespit eden fonksiyonlar.
* **Fonksiyonlar**:
  + def validate\_proxy(proxy): - Proxy’nin geçerliliğini kontrol eder.
  + def test\_proxies(proxies): - Bir dizi proxy’yi test eder.

**user\_agent\_manager/**

**1. user\_agent\_manager.py**

* **Amacı**: Kullanıcı ajanı yönetimini merkezi hale getirir.
* **İçerik**:
  + Kullanıcı ajanları listesi.
* **Fonksiyonlar**:
  + def get\_user\_agent(): - Rastgele bir kullanıcı ajanı döner.

**2. user\_agent\_loader.py**

* **Amacı**: Kullanıcı ajanlarını yükler.
* **İçerik**:
  + Kullanıcı ajanı yükleme işlemleri.
* **Fonksiyonlar**:
  + def load\_user\_agents\_from\_file(file\_path): - Dosyadan kullanıcı ajanı yükler.
  + def load\_user\_agents\_from\_api(api\_url): - API’den kullanıcı ajanı yükler.

**logging\_manager/**

**1. logging\_manager.py**

* **Amacı**: Projeye genel bir loglama mekanizması sağlar.
* **İçerik**:
  + Loglama düzeyleri ve yapılandırmaları.
* **Fonksiyonlar**:
  + def setup\_logging(log\_file): - Loglama ayarlarını yapar.
  + def log\_message(level, message): - Mesajı belirtilen düzeyde loglar.

**exceptions/**

**1. scraper\_exceptions.py**

* **Amacı**: Projede kullanılacak özel hata sınıflarını tanımlar.
* **İçerik**:
  + Hata sınıfları.
* **Fonksiyonlar**:
  + class ScraperException(Exception): - Genel bir özel hata.
  + class ProxyError(ScraperException): - Proxy hataları için özel sınıf.
  + class UserAgentError(ScraperException): - Kullanıcı ajanı hataları için özel sınıf.

**tests/**

**1. \_\_init\_\_.py**

* **Amacı**: Test modülünün bir Python paketi olarak tanınmasını sağlar.
* **İçerik**: Genellikle boş bırakılır.

**2. test\_scraper.py**

* **Amacı**: Scraper’ın ana işlevselliğini test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_fetch\_data(): - Veri çekme işlevini test eder.
  + def test\_parse\_response(): - Yanıtın doğru şekilde işlendiğini test eder.

**3. test\_data\_parser.py**

* **Amacı**: Verilerin işlenmesiyle ilgili fonksiyonları test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_save\_data(): - Verinin kaydedildiğini test eder.
  + def test\_process\_data(): - İşleme fonksiyonunu test eder.

**4. test\_proxy\_manager.py**

* **Amacı**: Proxy yönetimi ile ilgili işlevleri test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_manage\_proxies(): - Proxy yönetim fonksiyonunu test eder.
  + def test\_validate\_proxy(): - Proxy geçerliliğini kontrol eden fonksiyonu test eder.

**5. test\_user\_agent\_manager.py**

* **Amacı**: Kullanıcı ajanı yönetimi ile ilgili işlevleri test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_get\_user\_agent(): - Kullanıcı ajanı alma fonksiyonunu test eder.

**6. test\_logging\_manager.py**

* **Amacı**: Loglama işlemlerinin doğruluğunu test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_log\_message(): - Log mesajının doğru bir şekilde yazıldığını test eder.

**7. test\_utils.py**

* **Amacı**: Yardımcı fonksiyonların işlevselliğini test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_send\_request(): - HTTP istek fonksiyonunu test eder.
  + def test\_sleep\_random(): - Rastgele bekleme fonksiyonunu test eder.

**8. test\_site\_detector.py**

* **Amacı**: Site tespit işlevlerinin doğruluğunu test etmek için kullanılır.
* **Fonksiyonlar**:
  + def test\_detect\_site\_structure(): - Site tespit fonksiyonunu test eder.

**logs/**

**1. scraper.log**

* **Amacı**: Kazıma işlemleri sırasında oluşan tüm log mesajlarını içerir.
* **İçerik**: Loglama bilgileri, hata mesajları ve diğer geri bildirimler.

**data/**

**1. all\_listings.csv**

* **Amacı**: Kazıma sonucunda elde edilen tüm verilerin saklandığı dosya.
* **İçerik**: Ürün bilgileri (ad, fiyat, açıklama vb.).

**2. raw\_data/**

* **Amacı**: Kazıma sırasında elde edilen ham HTML yanıtlarını saklar.
* **İçerik**: HTML dosyaları.

**3. processed\_data/**

* **Amacı**: İşlenmiş ve temizlenmiş verilerin saklandığı klasör.
* **İçerik**: İşlenmiş veri dosyaları.

**4. proxies.txt**

* **Amacı**: Proxy sunucularının listelendiği dosya.
* **İçerik**: Proxy adresleri.

**5. user\_agents.txt**

* **Amacı**: Kullanıcı ajanlarının listelendiği dosya.
* **İçerik**: Kullanıcı ajanı bilgileri.

**requirements.txt**

* **Amacı**: Projeyi çalıştırmak için gerekli Python kütüphanelerinin listelendiği dosya.
* **İçerik**: Gerekli kütüphanelerin adları ve sürüm numaraları.

**README.md**

* **Amacı**: Projenin tanıtımını, nasıl kullanılacağını ve yapılandırılacağını açıklar.
* **İçerik**: Proje hakkında bilgi, kullanım talimatları, katkı yönergeleri.

**CONTRIBUTING.md**

* **Amacı**: Projeye katkıda bulunmak isteyenler için yönergeleri içerir.
* **İçerik**: Katkı kuralları ve beklentileri.

**LICENSE**

* **Amacı**: Projenin lisans bilgilerini belirtir.
* **İçerik**: Kullanım ve dağıtım koşulları.

**.gitignore**

* **Amacı**: Versiyon kontrolünde göz ardı edilmesi gereken dosya ve dizinlerin listesini içerir.
* **İçerik**: Geçici dosyalar, log dosyaları ve kişisel ayar dosyaları.