```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time

class EbayScraper:
    def __init__(self, main_url, header):
        # Constructor to initialize the main_url and header
        self.main_url = main_url
        self.header = header

    def content_scraper(self, urls, ratings, ratings_count, prices, pn):
        # Method to scrape content from product URLs
        condition = []
        seller_note = []
        processor = []
        screen_size = []
        manufacturer_color = []
        color = []
        ram_size = []
        ssd_capacity = []
        gpu = []
        processor_speed = []
        brand = []
        type_ = []
        release_year = []
        maximum_resolution = []
        model = []
        os = []
        features = []
        hard_drive_capacity = []
        country_region_of_manufacture = []
        storage_type = []

        for u in urls:
            r = requests.get(u, headers=self.header)
            soup = BeautifulSoup(r.content, 'html.parser')
            feature = soup.find_all('div', class_='ux-layout-section-evo__col')
            features_dictionary = {}

            if len(feature) == 0:
                f = soup.find('div', class_='description')
                if f is None:
                    # Extract features from the description section
                    condition.append(features_dictionary.get('Condition'))
                    seller_note.append(features_dictionary.get('Seller Notes'))
                    processor.append(features_dictionary.get('Processor'))
                    screen_size.append(features_dictionary.get('Screen Size'))
                    manufacturer_color.append(features_dictionary.get('Manufacturer Color'))
                    color.append(features_dictionary.get('Color'))
                    ram_size.append(features_dictionary.get('RAM Size'))
                    ssd_capacity.append(features_dictionary.get('SSD Capacity'))
                    gpu.append(features_dictionary.get('GPU'))
                    processor_speed.append(features_dictionary.get('Processor Speed'))
                    brand.append(features_dictionary.get('Brand'))
                    type_.append(features_dictionary.get('Type'))
                    release_year.append(features_dictionary.get('Release Year'))
                    maximum_resolution.append(features_dictionary.get('Maximum Resolution'))
                    model.append(features_dictionary.get('Model'))
                    os.append(features_dictionary.get('Operating System'))
                    features.append(features_dictionary.get('Features'))
                    hard_drive_capacity.append(features_dictionary.get('Hard Drive Capacity'))
                    country_region_of_manufacture.append(features_dictionary.get('Country/Region of Manufacture'))
                    storage_type.append(features_dictionary.get('Storage Type'))

                else:
                    # Extract features from the list items in the description
                    for i in f.find_all('li'):
                        if i.find('div', class_='s-name') == None:
                            pass
                        else:
                            label = i.find('div', class_='s-name').text
                            value = i.find('div', class_='s-value').text

                            if label == 'Condition':
```

```
                        features_dictionary['Condition'] = value
                    elif label == 'Seller Notes':
                        features_dictionary['Seller Notes'] = value
                    elif label == 'Processor':
                        features_dictionary['Processor'] = value
                    elif label == 'Screen Size':
                        features_dictionary['Screen Size'] = value
                    elif label == 'Manufacturer Color':
                        features_dictionary['Manufacturer Color'] = value
                    elif label == 'Color':
                        features_dictionary['Color'] = value
                    elif label == 'RAM Size':
                        features_dictionary['RAM Size'] = value
                    elif label == 'SSD Capacity':
                        features_dictionary['SSD Capacity'] = value
                    elif label == 'GPU':
                        features_dictionary['GPU'] = value
                    elif label == 'Processor Speed':
                        features_dictionary['Processor Speed'] = value
                    elif label == 'Brand':
                        features_dictionary['Brand'] = value
                    elif label == 'Type':
                        features_dictionary['Type'] = value
                    elif label == 'Release Year':
                        features_dictionary['Release Year'] = value
                    elif label == 'Maximum Resolution':
                        features_dictionary['Maximum Resolution'] = value
                    elif label == 'Model':
                        features_dictionary['Model'] = value
                    elif label == 'Operating System':
                        features_dictionary['Operating System'] = value
                    elif label == 'Features':
                        features_dictionary['Features'] = value
                    elif label == 'Hard Drive Capacity':
                        features_dictionary['Hard Drive Capacity'] = value
                    elif label == 'Country/Region of Manufacture':
                        features_dictionary['Country/Region of Manufacture'] = value
                    elif label == 'Storage Type':
                        features_dictionary['Storage Type'] = value
            time.sleep(0.2)

    else:
        # Extract features from the labeled values in the page
        for i in feature:
            if i.find('div', class_='ux-labels-values__labels') == None:
                pass
            else:
                label = i.find('div', class_='ux-labels-values__labels').text
                value = i.find('div', class_='ux-labels-values__values').text

                if label == 'Condition':
                    features_dictionary['Condition'] = value
                elif label == 'Seller Notes':
                    features_dictionary['Seller Notes'] = value
                elif label == 'Processor':
                    features_dictionary['Processor'] = value
                elif label == 'Screen Size':
                    features_dictionary['Screen Size'] = value
                elif label == 'Manufacturer Color':
                    features_dictionary['Manufacturer Color'] = value
                elif label == 'Color':
                    features_dictionary['Color'] = value
                elif label == 'RAM Size':
                    features_dictionary['RAM Size'] = value
                elif label == 'SSD Capacity':
                    features_dictionary['SSD Capacity'] = value
                elif label == 'GPU':
                    features_dictionary['GPU'] = value
                elif label == 'Processor Speed':
                    features_dictionary['Processor Speed'] = value
                elif label == 'Brand':
                    features_dictionary['Brand'] = value
                elif label == 'Type':
                    features_dictionary['Type'] = value
                elif label == 'Release Year':
```

```
                        features_dictionary['Release Year'] = value
                    elif label == 'Maximum Resolution':
                        features_dictionary['Maximum Resolution'] = value
                    elif label == 'Model':
                        features_dictionary['Model'] = value
                    elif label == 'Operating System':
                        features_dictionary['Operating System'] = value
                    elif label == 'Features':
                        features_dictionary['Features'] = value
                    elif label == 'Hard Drive Capacity':
                        features_dictionary['Hard Drive Capacity'] = value
                    elif label == 'Country/Region of Manufacture':
                        features_dictionary['Country/Region of Manufacture'] = value
                    elif label == 'Storage Type':
                        features_dictionary['Storage Type'] = value

                time.sleep(0.2)

            # Append features to respective lists
            condition.append(features_dictionary.get('Condition'))
            seller_note.append(features_dictionary.get('Seller Notes'))
            processor.append(features_dictionary.get('Processor'))
            screen_size.append(features_dictionary.get('Screen Size'))
            manufacturer_color.append(features_dictionary.get('Manufacturer Color'))
            color.append(features_dictionary.get('Color'))
            ram_size.append(features_dictionary.get('RAM Size'))
            ssd_capacity.append(features_dictionary.get('SSD Capacity'))
            gpu.append(features_dictionary.get('GPU'))
            processor_speed.append(features_dictionary.get('Processor Speed'))
            brand.append(features_dictionary.get('Brand'))
            type_.append(features_dictionary.get('Type'))
            release_year.append(features_dictionary.get('Release Year'))
            maximum_resolution.append(features_dictionary.get('Maximum Resolution'))
            model.append(features_dictionary.get('Model'))
            os.append(features_dictionary.get('Operating System'))
            features.append(features_dictionary.get('Features'))
            hard_drive_capacity.append(features_dictionary.get('Hard Drive Capacity'))
            country_region_of_manufacture.append(features_dictionary.get('Country/Region of Manufacture'))
            storage_type.append(features_dictionary.get('Storage Type'))

        time.sleep(1)
        pd.DataFrame({
            'Product Url': urls,
            'Brand': brand,
            'Price': prices,
            'Rating': ratings,
            'Ratings Count': ratings_count,
            'Condition': condition,
            'Seller Note': seller_note,
            'Processor': processor,
            'Screen Size': screen_size,
            'Manufacturer Color': manufacturer_color,
            'Color': color,
            'Ram Size': ram_size,
            'SSD Capacity': ssd_capacity,
            'GPU': gpu,
            'Processor Speed': processor_speed,
            'Type': type_,
            'Release Year': release_year,
            'Maximum Resolution': maximum_resolution,
            'Model': model,
            'OS': os,
            'Features': features,
            'Hard Drive Capacity': hard_drive_capacity,
            'Country Region Of Manufacture': country_region_of_manufacture,
            'Storage Type': storage_type
        }).to_csv(f'PC Laptops & Netbooks (Product Data)/product_data{pn}.csv',
                  index=False)
        print(f'Data in page: {pn} scraped.')

    def url_scraper(self, pages_n):
        # Method to scrape product URLs from eBay search pages
        for pn in range(1, pages_n):
            product_urls = []
            ratings = []
```

```
            ratings_count = []
            prices = []

            # Construct the URL for the current page
            u = self.main_url[:-1] + str(pn)

            # Make a request to the search page
            r = requests.get(u, headers=self.header)
            if r.status_code == 200:
                soup = BeautifulSoup(r.content, 'html.parser')

                # Find the list of products on the page
                main_page = soup.find('ul', class_='b-list__items_nofooter')
                products = main_page.find_all('li', class_='s-item s-item--large')

                # Iterate through products on the page
                for i in products:
                    # Extract product details such as URL, price, rating, and ratings count
                    link = i.find('a', class_='s-item__link')
                    product_urls.append(link.get('href'))

                    try:
                        prices.append(i.find('span', class_='s-item__price').text)
                    except:
                        prices.append('')

                    if i.find('div', class_='star-rating b-rating__rating-star'):
                        r = i.find('div', class_='star-rating b-rating__rating-star').get('aria-label')

                        r_c = i.find('span', class_='b-rating__rating-count').text
                        start_idx = r_c.find('(')
                        end_idx = r_c.find(')')

                        ratings_count.append(r_c[start_idx + 1:end_idx])

                        ratings.append(r)
                    else:
                        ratings.append('')
                        ratings_count.append('')

                # Create a DataFrame and save it to a CSV file with product URLs
                pd.DataFrame({'Product Url': product_urls, 'Rating': ratings, 'Ratings Count': ratings_count,
                              'Price': prices}).to_csv(
                    f'PC Laptops & Netbooks (Product Url)/producturl_{pn}.csv',
                    index=False)

                print(f'Urls in page: {pn} scraped.')
                time.sleep(3)

                # Call content_scraper to scrape detailed information from each product URL
                self.content_scraper(product_urls, ratings, ratings_count, prices, pn)
            else:
                print(r.status_code)
                break

# Example usage:
# main_url = "https://www.ebay.com/sch/i.html?_nkw=laptops&_pgn="
# header = {"User-Agent": "Your User Agent"}
# ebay_scraper = EbayScraper(main_url, header)
# ebay_scraper.url_scraper(5)  # Scrape information from the first 5 pages of search results
```