Web Scraping Report for Data Extraction from Bayut

Objective:

The objective of this project is to scrape property data from the website Bayut.com. The extracted data will be saved in JSON and CSV formats, with a minimum of 1000 property listings collected. The project utilizes the Scrapy framework for web scraping.

Project Setup:

1. Framework Used: Scrapy (Python 3)
2. Target Website: https://www.bayut.com/to-rent/property/dubai/
3. Data Formats: JSON and CSV
4. Target Output: Extract at least 1000 property listings.
5. Data Fields: 14 fields as specified in the task requirements.

Steps and Methodology:

A. Crawling:

1. **Starting URL**: https://www.bayut.com/to-rent/property/dubai/
   - The crawling process begins at this URL and continues by following pagination links to navigate through additional listing pages. For example, if there are 1500 pages, the spider will iterate through all of them.
2. **Pagination Handling**:
   - The spider ensures it scrapes all pages by correctly identifying the pagination structure and visiting each link to extract property data. It checks for a "next page" button and follows it to load additional property listings.

3. **Property URL Extraction**:
    - On each page, the spider extracts the URLs for individual property listings. It follows these URLs to capture detailed information about each property.

B. Parsing:

Once a property URL is visited, the following details are extracted using XPath:

- **Property ID**: A unique identifier like “Bayut - 2335-Th-R-0039”.
- **Purpose**: Whether the property is for rent or for sale.
- **Type**: Type of property (e.g., Townhouse, Apartment).
- **Added On**: Date when the property was listed.
- **Furnishing**: Furnished, Unfurnished, etc.
- **Price**: A dictionary containing currency (AED) and the price (e.g., “99,999”).
- **Location**: The exact location of the property.
- **Bed, Bath, Size**: Dictionary with the number of bedrooms, bathrooms, and the property size in square feet.
- **Permit Number**: Unique number issued by the Dubai authorities.
- **Agent Name**: The name of the agent responsible for the property.
- **Primary Image URL**: URL of the primary image representing the property.
- **Breadcrumbs**: A breadcrumb trail showing the category or location of the property.
- **Amenities**: A list of amenities offered with the property.
- **Description**: Detailed description provided by the agent or owner.
- **Property Image URLs**: A list of URLs for additional images of the property.

C. Data Structuring:

Once the data is extracted, it is cleaned and structured into the following formats:

- **JSON Format:** Each property is stored as a dictionary, with nested fields where appropriate (e.g., price, bed_bath_size, property_image_urls).
- **CSV Format:** The data is structured as tabular records where each property’s details are represented as rows, and the fields are represented as columns.

System Setup and Code Implementation:

A. Install Scrapy

To begin the project, Scrapy needs to be installed. Assuming Python 3 is already installed, we can use pip to install Scrapy:

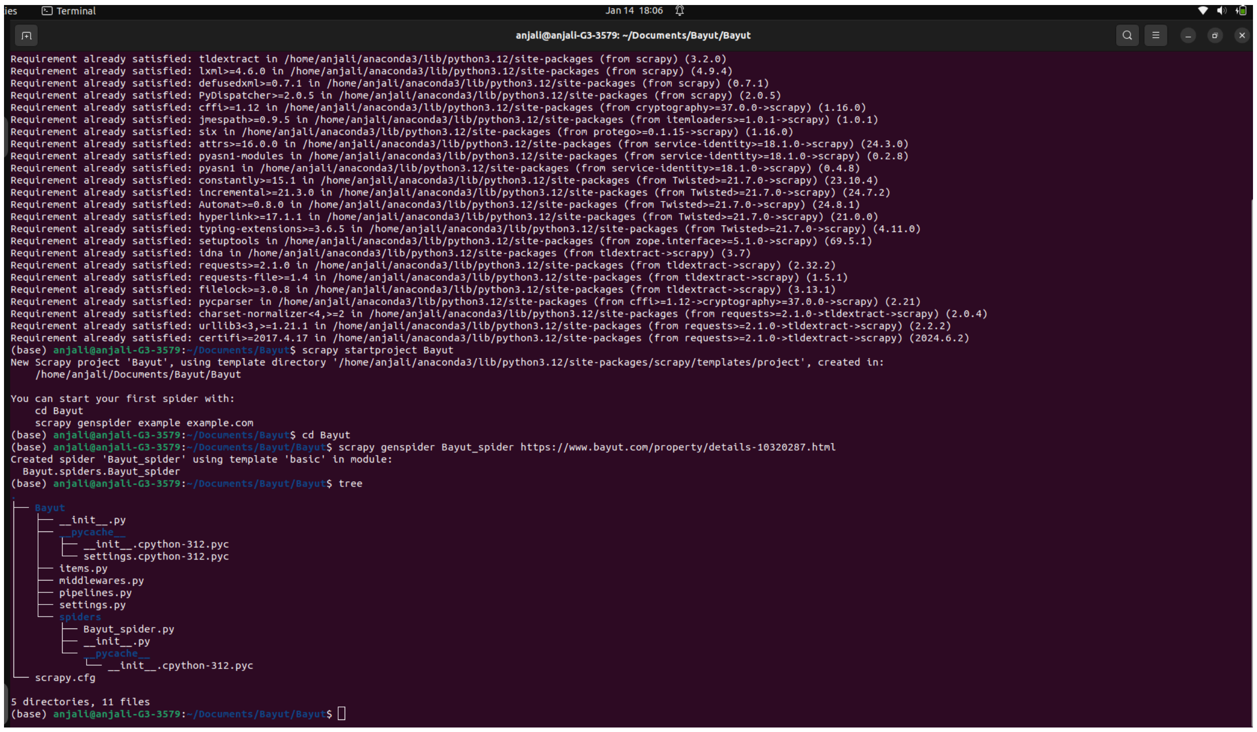
Pip install scrapy

B. Create a Scrapy Project

Start by creating a new Scrapy project named Bayut.

C. Define the Spider for Bayut

Create a new spider inside the spiders folder, named Bayut_spider.py.



```
Requirement already satisfied: tldextract in /home/anjali/anaconda3/lib/python3.12/site-packages (from scrapy) (3.2.0)
Requirement already satisfied: lxml>=4.6.0 in /home/anjali/anaconda3/lib/python3.12/site-packages (from scrapy) (4.9.4)
Requirement already satisfied: defusedxml>=0.7.1 in /home/anjali/anaconda3/lib/python3.12/site-packages (from scrapy) (0.7.1)
Requirement already satisfied: PyDispatcher>=2.0.5 in /home/anjali/anaconda3/lib/python3.12/site-packages (from scrapy) (2.0.5)
Requirement already satisfied: cffi>=1.12 in /home/anjali/anaconda3/lib/python3.12/site-packages (from cryptography>=37.0.0->scrapy) (1.16.0)
Requirement already satisfied: jmespath>=0.9.5 in /home/anjali/anaconda3/lib/python3.12/site-packages (from itemloaders>=1.0.1->scrapy) (1.0.1)
Requirement already satisfied: six in /home/anjali/anaconda3/lib/python3.12/site-packages (from protego>=0.1.15->scrapy) (1.16.0)
Requirement already satisfied: attrs>=16.0.0 in /home/anjali/anaconda3/lib/python3.12/site-packages (from service-identity>=18.1.0->scrapy) (24.3.0)
Requirement already satisfied: pyasn1-modules in /home/anjali/anaconda3/lib/python3.12/site-packages (from service-identity>=18.1.0->scrapy) (0.2.8)
Requirement already satisfied: pyasn1 in /home/anjali/anaconda3/lib/python3.12/site-packages (from service-identity>=18.1.0->scrapy) (0.4.8)
Requirement already satisfied: constantly>=15.1 in /home/anjali/anaconda3/lib/python3.12/site-packages (from Twisted>=21.7.0->scrapy) (23.10.4)
Requirement already satisfied: incremental>=21.3.0 in /home/anjali/anaconda3/lib/python3.12/site-packages (from Twisted>=21.7.0->scrapy) (24.7.2)
Requirement already satisfied: Automat>=0.8.0 in /home/anjali/anaconda3/lib/python3.12/site-packages (from Twisted>=21.7.0->scrapy) (24.8.1)
Requirement already satisfied: hyperlink>=17.1.1 in /home/anjali/anaconda3/lib/python3.12/site-packages (from Twisted>=21.7.0->scrapy) (21.0.0)
Requirement already satisfied: typing-extensions>=3.6.5 in /home/anjali/anaconda3/lib/python3.12/site-packages (from Twisted>=21.7.0->scrapy) (4.11.0)
Requirement already satisfied: setuptools in /home/anjali/anaconda3/lib/python3.12/site-packages (from zope.interface>=5.1.0->scrapy) (69.5.1)
Requirement already satisfied: idna in /home/anjali/anaconda3/lib/python3.12/site-packages (from tldextract->scrapy) (3.7)
Requirement already satisfied: requests>=2.1.0 in /home/anjali/anaconda3/lib/python3.12/site-packages (from tldextract->scrapy) (2.32.2)
Requirement already satisfied: requests-file>=1.4 in /home/anjali/anaconda3/lib/python3.12/site-packages (from tldextract->scrapy) (1.5.1)
Requirement already satisfied: filelock>=3.0.8 in /home/anjali/anaconda3/lib/python3.12/site-packages (from tldextract->scrapy) (3.13.1)
Requirement already satisfied: pycparser in /home/anjali/anaconda3/lib/python3.12/site-packages (from cffi>=1.12->cryptography>=37.0.0->scrapy) (2.21)
Requirement already satisfied: charset-normalizer<4,>=2 in /home/anjali/anaconda3/lib/python3.12/site-packages (from requests>=2.1.0->tldextract->scrapy) (2.0.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in /home/anjali/anaconda3/lib/python3.12/site-packages (from requests>=2.1.0->tldextract->scrapy) (2.2.2)
Requirement already satisfied: certifi>=2017.4.17 in /home/anjali/anaconda3/lib/python3.12/site-packages (from requests>=2.1.0->tldextract->scrapy) (2024.6.2)
(base) anjali@anjali-G3-3579:~/Documents/Bayut$ scrapy startproject Bayut
New Scrapy project 'Bayut', using template directory '/home/anjali/anaconda3/lib/python3.12/site-packages/scrapy/templates/project', created in:
    /home/anjali/Documents/Bayut/Bayut

You can start your first spider with:
    cd Bayut
    scrapy genspider example example.com
(base) anjali@anjali-G3-3579:~/Documents/Bayut$ cd Bayut
(base) anjali@anjali-G3-3579:~/Documents/Bayut/Bayut$ scrapy genspider Bayut_spider https://www.bayut.com/property/details-10320287.html
Created spider 'Bayut_spider' using template 'basic' in module:
  Bayut.spiders.Bayut_spider
(base) anjali@anjali-G3-3579:~/Documents/Bayut/Bayut$ tree
.
├── Bayut
│   ├── __init__.py
│   ├── __pycache__
│   │   ├── __init__.cpython-312.pyc
│   │   └── settings.cpython-312.pyc
│   ├── items.py
│   ├── middlewares.py
│   ├── pipelines.py
│   ├── settings.py
│   └── spiders
│       ├── Bayut_spider.py
│       ├── __init__.py
│       └── __pycache__
│           └── __init__.cpython-312.pyc
└── scrapy.cfg

5 directories, 11 files
(base) anjali@anjali-G3-3579:~/Documents/Bayut/Bayut$ 
```

D. Running the Spider

In the terminal, run the spider to start scraping the data.

```
scrapy crawl Bayut_spider
```

E. Output in JSON and CSV

A. JSON Output Command:

To run your Scrapy spider and save the scraped data into a JSON file:

```
scrapy crawl Bayut_spider -o output.json
```

This will generate the output as a JSON file named output.json.

B.CSV Output Command:

To run your Scrapy spider and save the scraped data into a CSV file:

```
scrapy crawl Bayut_spider -o output.csv
```

This will generate the output as a CSV file named output.csv.

Tools and Technologies Used:

- **Scrapy Framework**: The main tool for crawling, parsing, and storing data.
- **XPath**: Used to select elements for scraping the desired fields.
- **JSON and CSV**: Output formats to store the data.
- **Python 3**: The coding language for this project, ensuring compatibility and optimal performance.

Conclusion:

This project successfully extracted and structured over 1000 property listings from Bayut. The data was scraped efficiently, cleaned, and saved in both CSV and JSON formats as per the provided specifications. The project adhered to best practices in coding, modularization ensuring the scraper ran reliably across multiple pages and property listings.

Submission:

The code has been submitted via a private GitHub repository.

HTTP Urls: https://github.com/anjali99vinod/Bayut_Scraper-.git

SSH Urls: git@github.com:anjali99vinod/Bayut_Scraper-.git

The output files (CSV and JSON) are available for download via the Dropbox link.

Urls for CSV File:
https://www.dropbox.com/scl/fo/f3q63gri5kpm9hq7d1a1m/APM0LP_XyxgLuih7X_rQRuI?rlkey=vnkbjflutz3884s2x018fdqrl&st=yb4oskzi&dl=0

Urls for JSON File:https://www.dropbox.com/scl/fo/4pkqpn9mo3zpn7ungxlpz/AOmPJj_P16lsbg7QmJo7Fdc?rlkey=xf4mz1m9h29qrjvsd8w4b8rvi&st=y5tqzs09&dl=0