Vehicle Auction Data Scraper

1. Approach

The script automates the extraction of vehicle data from an auction website using Selenium to navigate and locate relevant information. It extracts vehicle names, and stock numbers, and downloads images, saving them in organized folders. Error handling is in place to ensure smooth execution, and the data is stored in a CSV file for easy reference. The code is designed to be scalable and modular for future enhancements.

2. Workflow

2.1 Main Function (`main`)

This is the entry point for the script:

1. **Web Browser Initialization**:
   The script initializes an undetectable Chrome browser (`uc.Chrome`) to navigate the auction website.
2. **Navigating to the Auction Page**:
   The script opens the auction website URL and waits for it to load.
3. **Finding Auctions**:
   The script searches for auction listings on the page. If it finds one, it clicks on the first auction's calendar icon to open the auction's detail page.
4. **Calling `get_auction_data` Function**:
   Once the auction page is loaded, the script calls the `get_auction_data()` function to extract vehicle information and images.
5. Parameter
   i. **Proxy_file_name**: proxy ip txt file path
   ii. **Store_data_file_nam**e: scrape data stored in this file

iii. **Output_folder:** Save the scraped images and CSV folder path
iv. **Script_start_with_proxy:** start python scraping scripts with proxy ip(boolean)

2.2 Vehicle Data Extraction (`get_auction_data`)

1. **Extracting Vehicle Information**:
    - The script locates all vehicles listed in the auction and loops through each one.
    - For each vehicle, it extracts:
        - **Vehicle Name**: Captured from the vehicle name link.
        - **Stock Number**: Extracted from a field containing the stock number.
    - The extracted information is stored in a list of dictionaries for future use.
    - Parameter
        1. **Driver**: Selenium WebDriver instance used to navigate the auction website.
        2. **Store_data_file_name:** Store scrape data in this file.
        3. **Output_folder:** Save the scraped images and CSV folder path
        4. **Script_start_with_proxy:** Use this flag to enable proxy IP usage in the script.
2. **Image Download**:
    - The script navigates to each vehicle's detailed page by clicking on the vehicle's name.
    - Once on the detail page, it locates all images displayed in a specific image container.
    - **Downloading Images**:
      Each image is downloaded and saved in the appropriate vehicle folder created earlier.
    - **Image Processing**:
      The script reads the downloaded images, crops them (removes 10 pixels from the bottom), and then saves the cropped images.
3. **Back to Auction List**:
   After processing each vehicle, the script navigates back to the auction list to process the next vehicle.

2.3 Proxy Selection Function (select_proxy)

The `select_proxy` function is designed to load and randomly select a proxy IP address from a specified text file.

Parameters:

- `file_name`: The name of the text file containing a list of proxy IP addresses.

3. Data Storage

1. **Vehicle Data Storage**:
   - The vehicle name and stock number data are saved in a CSV file named `vehicle_data.csv`. This allows the extracted data to be stored and referenced later.
2. **Image Storage**:
   - The vehicle images are saved in a structured folder system:
     - **Root Folder**: `vehicle_images/`
     - **Subfolders**: One folder per vehicle, named using the vehicle name (with spaces replaced by underscores).

4. Setup Instructions:

- Set up a virtual environment and install the dependencies listed in the requirements.txt file.
- In the scraping.py file, configure the paths for the proxy_file_name and store_data_file_name files.
- Create a proxy IP text file and insert some proxy IPs into it.
- Execute python file
    - python scraping.py --proxy_file proxy_ip.txt --store_data_file vehicle_data.csv --output_folder output --script_start_with_proxy False
        - Proxy_file: Path to the proxy IP text file.
        - Store_data_file: Path to the output CSV file for storing data.
        - Output_folder: Path to the output folder.
        - Script_start_with_proxy:

            Path to the file containing proxy IPs.If running the script with a proxy, set use_proxy to True. Note that using a proxy may slow down the process.