

Project Presentation: Auto Market Scraper & Dashboard

What is this project?

It is an application created in Python that: automatically searches car sale ads from Facebook Marketplace and Craigslist, saves the results, and displays them in an interactive control panel, where the user can filter, sort and analyze the data quickly.

What do you need to do to start the application?

Have Python installed (version 3.9+). Click on the file START_CLICK_HERE.bat That's it. The rest is handled by the application: Automatically installs all necessary libraries. Starts the server. Automatically opens the browser with the control panel.

The screenshot shows the 'Scraper Control Panel' interface. It is divided into two main sections: 'Facebook' on the left and 'Craigslist' on the right. Each section has a 'How it works' tab and a 'Save-Set Config' tab. Under 'Save-Set Config', there are dropdown menus for 'State' and 'Price Min/Max' (Facebook) or 'Year Min/Max' and 'Miles Min/Max' (Craigslist). Below these are 'Save New Config Search' and 'Generate YAML' buttons. Under 'Generate YAML', there are buttons for 'Generate YAML Facebook' (Facebook) and 'Generate YAML Craigslist' (Craigslist). Below these are status indicators: 'Scrapers: Stopped' (red), 'Extractor: Stopped' (red), and 'Display: Running' (green). At the bottom are buttons to 'Start Scraper', 'Stop Scraper', 'Start Extractor', 'Stop Display', and 'Stop Extractor'.

What does the application contain?

Structure:
auto_market_v1 – the part that extracts the data
display – the part that displays the data
templates – the graphical web interface
dashboard.py – the main server
START_CLICK_HERE.bat – the launcher file

Name	Date Modified	Type	Size
auto_market_v1	6/23/2025 7:47 PM	File folder	
display	6/22/2025 5:59 PM	File folder	
templates	6/22/2025 5:59 PM	File folder	
dashboard	6/22/2025 5:53 PM	Python Source...	9 KB
START_CLICK_HERE	6/4/2025 10:16 PM	Windows Batc...	1 KB

How does the control panel look and what does it do?

A modern, easy-to-use web interface.
Buttons for starting/stopping Scraper, Extractor and Display.
Quick setup: choose the state, minimum and maximum price, car year, etc.
You can automatically generate YAML files for each search.

Auto Dashboard (Car Display)

After the data is extracted, you see it in a large, interactive table.
You can filter by anything: year, price, brand, city, fuel type, score, VIN, etc.
You also have buttons like "Top 1", "Top 2", "Top 3" – for quick filtering.
Everything is live, the data loads automatically from the local server.

What does it bring extra?

It is a complete system, from extraction to display.
It works locally, without needing external databases or complex servers.
It is modularly built, easy to extend and adapt.+

Control Panel – Full Control Over Searches

The panel is divided into two areas:

Facebook

You choose the state, minimum and maximum price.
Click “Save New Config Search” to save the settings.
Then generate the YAML configuration file.

Why only price filter?

Because Facebook requires real login and human behavior.

The app uses Selenium and simulates a human:

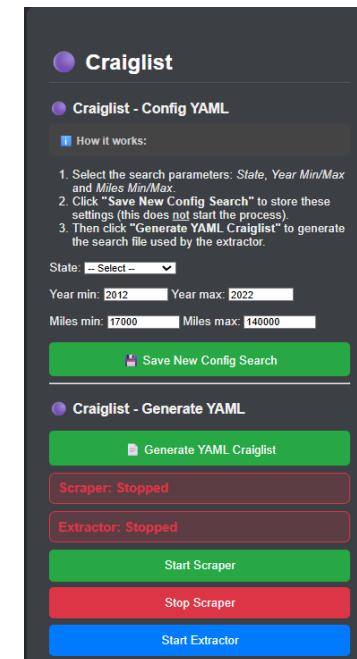
it fills filters using actual clicks,
scrolls the page to load new ads,
remembers already visited ads (smart logging),
avoids reloading the same data again and again.

We avoid complex queries that might get the account slowed down or blocked.



Craigslist

You can set: state, minimum and maximum year, minimum and maximum mileage.
No login is needed, so filtering is automatic and more flexible.
After setting, you generate the corresponding YAML file.



Why this approach?

This project was designed with responsibility and efficiency in mind:

For Facebook: a human-like, realistic, safe approach

For Craigslist: fast, direct, fully filterable scraping

We avoided overcomplicating the app needlessly.

The goal: reliability, efficiency, and full control without risk or account restrictions.

Smart Components

Logs for each search

Memory of already visited ads – they won't be checked again

Start/Stop system for every module:

Scraper

Extractor

Display

Automated, but manually controlled

The user has full control:

What filters to use

When to start or stop the process

What data to display

Explanation of Control Panel Buttons

After you configure your search (state, price, year, etc.), you have a set of control buttons that allow you to run the entire system precisely:

State & Minimum Price Selection (Facebook & Craigslist)

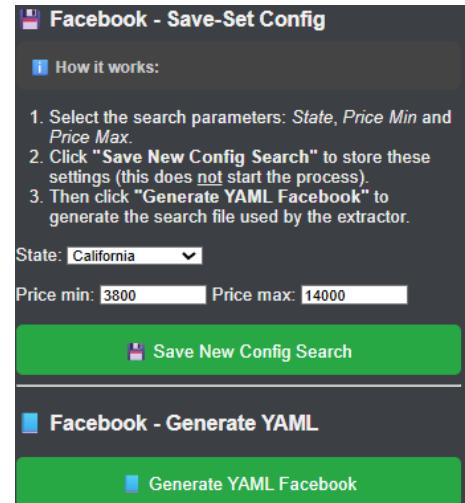
The app contains all U.S. states and cities.

When you select a state and a minimum price, that state is placed first in the scan list.

The program starts scanning with that state, then continues with all others, in order.

The price filter is applied to all states.

On Craigslist, it works the same way, but filters also include year and mileage.



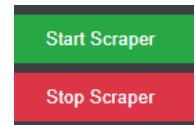
Start / Stop Scraper

Start Scraper

Starts the scraping program using the current YAML configuration file.

If you changed filters and clicked “Generate YAML”, the scraper will follow the new rules.

If not, it will use the existing YAML data.



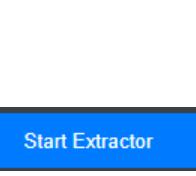
Stop Scraper

To stop the scraper safely, there's a well-thought-out internal mechanism:

When the scraper starts, it saves its process ID (PID) to a file.

When you press Stop, a separate program reads the PID and shuts down only the related processes.

The rest of the application (dashboard, extractor, display) keeps running.



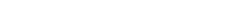
Start Extractor

This component processes all collected data.

It reads the raw data saved by the scraper and converts it into files ready to be displayed in the dashboard.

Start Extractor

When processing is done, the CMD window closes automatically, to avoid clutter.



Start Display

Opens a separate browser window at 127.0.0.1:5005.

Start Display

Here you'll see all collected listings in an interactive table.



The dashboard includes:

Advanced filters (year, price, brand, city, etc.)

Quick filter buttons: “Top 1”, “Top 2”, “Top 3”

Instant search and sorting

All key details about each vehicle

The Complete Flow in One Logical Sequence

Press Start Scraper → searches and collects listings

Press Start Extractor → processes and prepares the data

Press Start Display → opens the visual dashboard with all results

Dashboard – Smart & Interactive Display

Car Dashboard

Show per page: 100

Year Min	Mileage Min	Price Min	Odometer Min	Rating Min	Since Min	Safety Min	Clean Min	Has VIN Min	Options Plus	Condition	Top 1	Top 2	Top 3
2005	10000	\$1000	100000	1	2005-01-01	10000	100000	1000000000					
2010	15000	\$1500	150000	1	2010-01-01	15000	150000	1500000000					
2015	20000	\$2000	200000	1	2015-01-01	20000	200000	2000000000					
2020	25000	\$2500	250000	1	2020-01-01	25000	250000	2500000000					
2025	30000	\$3000	300000	1	2025-01-01	30000	300000	3000000000					
2030	35000	\$3500	350000	1	2030-01-01	35000	350000	3500000000					
2035	40000	\$4000	400000	1	2035-01-01	40000	400000	4000000000					
2040	45000	\$4500	450000	1	2040-01-01	45000	450000	4500000000					
2045	50000	\$5000	500000	1	2045-01-01	50000	500000	5000000000					
2050	55000	\$5500	550000	1	2050-01-01	55000	550000	5500000000					
2055	60000	\$6000	600000	1	2055-01-01	60000	600000	6000000000					
2060	65000	\$6500	650000	1	2060-01-01	65000	650000	6500000000					
2065	70000	\$7000	700000	1	2065-01-01	70000	700000	7000000000					
2070	75000	\$7500	750000	1	2070-01-01	75000	750000	7500000000					
2075	80000	\$8000	800000	1	2075-01-01	80000	800000	8000000000					
2080	85000	\$8500	850000	1	2080-01-01	85000	850000	8500000000					
2085	90000	\$9000	900000	1	2085-01-01	90000	900000	9000000000					
2090	95000	\$9500	950000	1	2090-01-01	95000	950000	9500000000					
2095	100000	\$10000	1000000	1	2095-01-01	100000	1000000	10000000000					
2100	105000	\$10500	1050000	1	2100-01-01	105000	1050000	10500000000					
2105	110000	\$11000	1100000	1	2105-01-01	110000	1100000	11000000000					
2110	115000	\$11500	1150000	1	2110-01-01	115000	1150000	11500000000					
2115	120000	\$12000	1200000	1	2115-01-01	120000	1200000	12000000000					
2120	125000	\$12500	1250000	1	2120-01-01	125000	1250000	12500000000					
2125	130000	\$13000	1300000	1	2125-01-01	130000	1300000	13000000000					
2130	135000	\$13500	1350000	1	2130-01-01	135000	1350000	13500000000					
2135	140000	\$14000	1400000	1	2135-01-01	140000	1400000	14000000000					
2140	145000	\$14500	1450000	1	2140-01-01	145000	1450000	14500000000					
2145	150000	\$15000	1500000	1	2145-01-01	150000	1500000	15000000000					
2150	155000	\$15500	1550000	1	2150-01-01	155000	1550000	15500000000					
2155	160000	\$16000	1600000	1	2155-01-01	160000	1600000	16000000000					
2160	165000	\$16500	1650000	1	2160-01-01	165000	1650000	16500000000					
2165	170000	\$17000	1700000	1	2165-01-01	170000	1700000	17000000000					
2170	175000	\$17500	1750000	1	2170-01-01	175000	1750000	17500000000					
2175	180000	\$18000	1800000	1	2175-01-01	180000	1800000	18000000000					
2180	185000	\$18500	1850000	1	2180-01-01	185000	1850000	18500000000					
2185	190000	\$19000	1900000	1	2185-01-01	190000	1900000	19000000000					
2190	195000	\$19500	1950000	1	2190-01-01	195000	1950000	19500000000					
2195	200000	\$20000	2000000	1	2195-01-01	200000	2000000	20000000000					
2200	205000	\$20500	2050000	1	2200-01-01	205000	2050000	20500000000					
2205	210000	\$21000	2100000	1	2205-01-01	210000	2100000	21000000000					
2210	215000	\$21500	2150000	1	2210-01-01	215000	2150000	21500000000					
2215	220000	\$22000	2200000	1	2215-01-01	220000	2200000	22000000000					
2220	225000	\$22500	2250000	1	2220-01-01	225000	2250000	22500000000					
2225	230000	\$23000	2300000	1	2225-01-01	230000	2300000	23000000000					
2230	235000	\$23500	2350000	1	2230-01-01	235000	2350000	23500000000					
2235	240000	\$24000	2400000	1	2235-01-01	240000	2400000	24000000000					
2240	245000	\$24500	2450000	1	2240-01-01	245000	2450000	24500000000					
2245	250000	\$25000	2500000	1	2245-01-01	250000	2500000	25000000000					
2250	255000	\$25500	2550000	1	2250-01-01	255000	2550000	25500000000					
2255	260000	\$26000	2600000	1	2255-01-01	260000	2600000	26000000000					
2260	265000	\$26500	2650000	1	2260-01-01	265000	2650000	26500000000					
2265	270000	\$27000	2700000	1	2265-01-01	270000	2700000	27000000000					
2270	275000	\$27500	2750000	1	2270-01-01	275000	2750000	27500000000					
2275	280000	\$28000	2800000	1	2275-01-01	280000	2800000	28000000000					
2280	285000	\$28500	2850000	1	2280-01-01	285000	2850000	28500000000					
2285	290000	\$29000	2900000	1	2285-01-01	290000	2900000	29000000000					
2290	295000	\$29500	2950000	1	2290-01-01	295000	2950000	29500000000					
2295	300000	\$30000	3000000	1	2295-01-01	300000	3000000	30000000000					
2300	305000	\$30500	3050000	1	2300-01-01	305000	3050000	30500000000					
2305	310000	\$31000	3100000	1	2305-01-01	310000	3100000	31000000000					
2310	315000	\$31500	3150000	1	2310-01-01	315000	3150000	31500000000					
2315	320000	\$32000	3200000	1	2315-01-01	320000	3200000	32000000000					
2320	325000	\$32500	3250000	1	2320-01-01	325000	3250000	32500000000					
2325	330000	\$33000	3300000	1	2325-01-01	330000	3300000	33000000000					
2330	335000	\$33500	3350000	1	2330-01-01	335000	3350000	33500000000					
2335	340000	\$34000	3400000	1	2335-01-01	340000	3400000	34000000000					
2340	345000	\$34500	3450000	1	2340-01-01	345000	3450000	34500000000					
2345	350000	\$35000	3500000	1	2345-01-01	350000	3500000	35000000000					
2350	355000	\$35500	3550000	1	2350-01-01	355000	3550000	35500000000					
2355	360000	\$36000	3600000	1	2355-01-01	360000	3600000	36000000000					
2360	365000	\$36500	3650000	1	2360-01-01	365000	3650000	36500000000					
2365	370000	\$37000	3700000	1	2365-01-01	370000	3700000	37000000000					
2370	375000	\$37500	3750000	1	2370-01-01	375000	3750000	37500000000					
2375	380000	\$38000	3800000	1	2375-01-01	380000	3800000	38000000000					
2380	385000	\$38500	3850000	1	2380-01-01	385000	3850000	38500000000					
2385	390000	\$39000	3900000	1	2385-01-01	390000	3900000	39000000000					
2390	395000	\$39500	3950000	1	2390-01-01	395000	3950000	39500000000					
2395	400000	\$40000	4000000	1	2395-01-01	400000	4000000	40000000000					
2400	405000	\$40500	4050000	1	2400-01-01	405000	4050000	40500000000					
2405	410000	\$41000	4100000	1	2405-01-01	410000	4100000	41000000000					
2410	415000	\$41500	4150000	1	2410-01-01	415000	4150000	41500000000					
2415	420000	\$42000	4200000	1	2415-01-01	420000	4200000	42000000000					
2420	425000	\$42500	4250000	1	2420-01-01	425000	4250000	42500000000					
2425	430000	\$43000	4300000	1	2425-01-01	430000	4300000	43000000000					
2430	435000	\$43500	4350000	1	2430-01-01	435000	4350000	43500000000					
2435	440000	\$44000	4400000	1	2435-01-01	440000	4400000	44000000000					
2440	445000	\$44500											

Year Min: 2013	Mileage Min: 0	Price Min: 4000	Own Min: 1	Rating Min: 0	Joined Min: Min	Since Min: Min	SScore Min: -2	Clean Min: 1	Has VIN Min: Min	Refresh Page	Top 1
Year Max: 2025	Mileage Max: 80000	Price Max: 17000	Own Max: 2	Rating Max: Max	Joined Max: Max	Since Max: Max	SScore Max: 10	Clean Max: 1	Has VIN Max: Max	Clear Filters	Top 2

[Top 1](#) 
[Top 2](#) 
[Top 3](#) 

Advanced Features

Filter by any column (top input, min/max values)

Top 1 / Top 2 / Top 3 buttons – quickly show the best-rated entries

Instant sorting – all columns are sortable

Direct link to ad – every row has an "Open Listing" button

Extracted comments – if the ad includes key phrases, they are displayed and interpreted

Why Is This System So Powerful?

Because it doesn't just extract raw data – it:

Understands, cleans, interprets, and structures it

Handles real ads with typos, informal expressions, and natural language

Displays everything in a clear, easy-to-navigate, and analytical dashboard

Connection to the Original Listing – Direct Contact with the Seller

One of the most important and useful features of the dashboard is that each row contains a direct link to the original ad, exactly as it was posted on:

Facebook Marketplace or Craigslist

How Does It Work?

In the last column (Marketplace), there is a button labeled "Open Listing."

When you click it, the real ad page opens in a new browser tab.

This allows you to contact the seller directly, without needing to search manually.

Has VIN	Marketplace
Yes	Open Listing
✓	Open Listing

Technology Used for This Dashboard

Tabulator.js – a powerful JavaScript library for building interactive tables

Custom JavaScript – used for:

managing filters and sorting,

dynamically generating links,

enabling real-time interaction with backend-processed data

Flask API + AJAX – connects Python data with the visual interface

Dynamic Refresh – no need to manually reload the page

The Result?

You get a user-friendly, fully interactive dashboard that lets you:

View all processed data clearly

Apply advanced filters instantly

Open the original listing with one click

Contact the seller in just a few seconds

Technologies Used in the Project

Backend – Core Application Logic

Technology Purpose

Python Main language used for backend logic and scripting.

Flask Lightweight web framework used for the control panel and API server.

YAML Configuration format for defining search criteria.

Selenium Browser automation (simulates human behavior on Facebook Marketplace).

Regex (re) Extracts complex data like VINs, years, statuses, and keyword patterns.

PID Handling Manages process IDs to control Start/Stop mechanisms reliably.

Logging Tracks visited listings and saved entries to avoid duplicates.

Data Processing – Extractor Module

Technology Purpose

Python Scripts Analyze raw scraped data.

Smart Filters Normalize incorrect labels (e.g., "cleean title") and detect insights.

Scoring Engine Generates auto-ratings based on listing quality, seller reputation, etc.

Frontend – User Interface & Interaction

Technology Purpose

HTML5 / CSS3 Web page structure and styling.

JavaScript (ES6) Interactive logic for dashboard and control buttons.

Tabulator.js High-performance interactive table (filtering, sorting, links).

AJAX (fetch) Loads data dynamically without refreshing the page.

Bootstrap (optional) Provides responsive design and modern layout (if used).

Networking & Interaction Flow

Component Role

localhost:5000 Control panel (Start/Stop system, generate configs).

localhost:5005 Interactive dashboard (view, filter, analyze listings).

Open Listing Link Direct link to original ad (Facebook or Craigslist).

Project Structure

Folder/File Description

auto_market_v1/ Scraper modules & YAML config system.

display/ Server handling dashboard interface.

templates/ HTML templates used in the dashboard.

dashboard.py Main Flask app entry point.

START_CLICK_HERE.bat Auto-launcher: installs requirements, starts app & browser.

Project Structure – auto_market_v1

Main Folders & Files (Core Components)

Name	Description
------	-------------

facebook/, craigslist/ Main scraping modules for Facebook Marketplace and Craigslist.

alfa_facebook.py, alfa_craigslist.py Main Python scripts for starting data extraction per platform.

fb_extract_json.py, craig_extract_json.py Data extractor modules that clean, validate, and prepare raw listings.

dashboard_facebook/, dashboard_craigslist/ Contains the logic and frontend files to display listings interactively via web dashboard.

fb_script_json/ Stores browser session/profile info (used for auto-login on Facebook).

logs/ Log files for monitoring sessions, useful for debugging and avoiding duplicates.

processed_pages/, saved_pages/ Stores raw and processed versions of the scraped HTML content.

chrome_profile/ Stores Chrome profile used by Selenium for automation with saved login.

session/ Temp session storage for ongoing scraping processes.

config/ Configuration folder with YAML files to define search parameters, filters, etc.

config.yaml, config_seller.yaml Specific YAML files used by the system to customize filtering and search logic.

craiglist_json, results.json Parsed and structured listing data, used by the dashboard frontend.

program_pid, craiglist_pid PID files used to track running Python processes (for clean Start/Stop).

install_and_run.bat Windows script for one-click install and launch.

First_run.py, login_first_time.py Scripts for setting up the system and logging into platforms for the first time.

README.md, INSTALL.md Documentation files explaining usage, install steps, and structure.

requirements.txt Lists all Python dependencies required to run the project.

Quick Summary

The system is built 100% in Python.

Combines Selenium, Regex, and smart filtering for high-quality data extraction.

Connects to a Flask-powered API that feeds a live interactive dashboard built with Tabulator.js.

Offers real-time data control, filtering, scoring, and direct links to live listings.

Final Note – Project Scope and Constraints

This software was designed to be local, compact, robust, and easy to launch, without relying on complex external services. Therefore:

Key Principles

Entirely written in pure Python.

Runs completely locally from a single folder.

No need for external databases or complicated installations.

No internet connection required for control or display functionality.

Accepted Limitations

To respect the minimalist and user-friendly architecture, a few trade-offs were accepted:

No dedicated domain for live online dashboard hosting.

No SQL-based database or live sync via remote APIs.

No standalone GUI application; everything is integrated and served locally.

Yet, despite these limitations, the application:

Works smoothly, with no crashes or bugs.

Provides a clean and functional interface.

Is highly useful for individuals or businesses looking to track car listings.

Stands as a reliable, real-world automation tool.

Respect for the Client & Development Ethics

This project is the result of a freelance collaboration, created at the request and with the support of a real client.

The source code will not be published publicly on GitHub.

It will not be distributed freely, out of respect for the client's vision, funding, and initiative.

Although the project was developed on a limited budget, it was treated with full seriousness and dedication.

In fact, more hours were invested than initially estimated — because:

“A respectful freelancer honors both their craft and the trust of the client.”

Personal Reflection – A Shift in Approach

This is the first project where I truly focused on practical use and real interaction with the end-user.

My Previous Projects:

Were often too technical, tailored for developers and IT professionals.

Sometimes hard to follow for non-technical users.

What Changed This Time?

From the beginning, this project was designed to be:

Easy to use.

Easy to understand.

Useful even for those with minimal technical knowledge.

The explanations were written as if talking to a car enthusiast, not just a programmer.

The real goal: To build not just functional software, but a logical, intuitive, and helpful tool.

Future Plans & Functionalities

This project was built modularly, with future improvements already in mind.

Database Integration (Internal + External)

Connection to a central database that:

Calculates average market price for each car model.

Enables intelligent price comparisons.

This will be the base for an automated alert system.

Smart Notification System (SMS / App)

If a profitable listing is detected (clean, under market value):

An alert will be sent automatically to the phone.

“Profitable” = Based on logical filtering:

No fake listings, no missing VIN, fair mileage, etc.

Clean, rare, undervalued listings get priority.

Improved Human Behavior Simulation

Enhance Selenium with:

Human-like delays, mouse movement, scroll simulation, etc.

Avoid unnecessary Facebook navigation ⇒ reduce risk of account ban.

Support for More Platforms

Add other car listing websites:

Autotrader, Cars.com, Mobile.de, OLX, and more.

Each integration will respect each site's structure and limits.

Editable & Smart Dashboard

Allow manual editing of any listing:

Status, price, notes, etc.

Mark modified data with a visual indicator (icon or color).

Add internal tracking:

Who modified a listing, what, and when.