1. Link Extractor

Purpose

Collect a clean list of every active Grays auction URL so downstream scrapers and dashboards always start from a consistent queue of listings.

Key capabilities

â¢ Runs ‘scripts/extract_links.py‘ when the **Run link scraper** button is pressed and surfaces immediate success / failure feedback in the UI.

â¢ Counts the rows in ‘CSV_data/all_vehicle_links.csv‘ and previews the first 20 links so you know what was captured in the latest crawl.

Data flow

â Input: none â the scraper hits Grays directly.

â Output: ‘CSV_data/all_vehicle_links.csv‘ (listing URL, title, metadata).

Notes

â Kick this page off whenever you want to refresh the entire discovery pipeline before running deeper scrapes.

2. Vehicle Detail Extractor

Purpose

Expand each tracked link into a fully structured record that captures specs, condition notes, pricing, and other static context for the Active Listings dashboard.

Key capabilities

â¢ Guards against missing link data â if ‘all_vehicle_links.csv‘ does not exist the page blocks you from running the detail scraper and surfaces an error state.

â¢ Runs ‘scripts/extract_vehicle_details.py‘, then previews up to 50 rows from ‘vehicle_static_details.csv‘ along with a total count so you can sanity-check the pull.

Data flow

â Input: ‘CSV_data/all_vehicle_links.csv‘ (required).

â Output: ‘CSV_data/vehicle_static_details.csv‘ (master spec sheet for each listing).

Notes

â Use this immediately after refreshing the link list so the remaining tools have up-to-date vehicle snapshots.

3. Active Listings Dashboard

Purpose

Serve as the mission control for live auctions: filter noisy stock, refresh bid data, and optionally trigger GPT-powered profit checks per vehicle.

Key capabilities

â¢ Loads ‘vehicle_static_details.csv‘, enforces that only ‘status == ’active’‘ records remain, and displays listing cards grouped by time-to-close buckets (<24h
â¢ 1-2d
â¢ 2-3d
â¢ 3+d).
â¢ Sidebar filters hide engine defect notes, unregistered vehicles, and/or anything outside Victoria so buyers can focus on viable stock.
â¢ Provides **Refresh Active Listings** and **Refresh Visible Listings** actions that call ‘scripts.update_bids.update_bids‘ (optionally limited to the filtered URLs).
â¢ Each card exposes a Run AI Analysis button. The handler sends the row through the OpenAI chat API, parses a JSON resale verdict, and persists the result in ‘CSV_data/ai_verdicts.csv‘ for future sessions.

Data flow

â Inputs: ‘vehicle_static_details.csv‘ plus optional ‘ai_verdicts.csv‘ for overlaying prior AI recommendations.
â Outputs: updated ‘vehicle_static_details.csv‘ (bid/time refreshes) and appended ‘ai_verdicts.csv‘ rows when new analyses run.

Notes

â Skipped URLs from bid refreshes are cached in-session so you can retry only the failures.

4. Master Database Overview

Purpose

Provide a quick audit of every lifecycle stage â active listings, sold stock, and referred vehicles â without jumping between CSVs.

Key capabilities

â¢ Verifies the presence of ‘vehicle_static_details.csv‘, ‘sold_cars.csv‘, and ‘referred_cars.csv‘ before rendering so you catch missing exports early.

â¢ Exposes an **Update Master Database** button that runs ‘scripts/update_master.py‘ and clears Streamlit caches so fresh data is immediately visible.

â¢ Uses a reusable renderer to show record counts plus up to 200-row previews for the Active, Sold, and Referred datasets (with the most relevant columns per table).

Data flow

â Inputs: the three CSV snapshots mentioned above.

â Output: refreshed CSVs when ‘update_master.py‘ is executed.

Notes

â Any missing columns are explicitly flagged so schema drift is easy to spot.

5. AI Pricing Analysis

Purpose

Blend rule-based pricing heuristics, historical sale comps, manual Carsales research, and GPT valuations to prioritise listings finishing soon.

Key capabilities

â¢ Requires a rich data stack (‘vehicle_static_details.csv‘, ‘active_vehicle_details.csv‘, ‘ai_verdicts.csv‘, ‘ai_listing_valuations.csv‘, and ‘sold_cars.csv‘) so comparisons always combine the latest auction context with historical baselines.

â¢ Lets you focus on a time window (24/48/72h) plus reuse the Active Listings filters to hide engine issues, unregistered stock, or non-VIC locations.

â¢ Calculates median discounts versus comparable sales, surfaces the most underpriced cars in one tab, and routes listings with no comps into a second review queue.

â¢ Inside each listing panel you can refresh bid data, run or re-run the Carsales-oriented GPT valuation (‘scripts.ai_listing_valuation.run_ai_listing_analysis‘), and capture manual Carsales research (instant offer, sell range, comps table, recent sales).

â¢ AI verdict widgets show Carsales estimate, recommended max bid, expected profit, and any qualitative confidence notes saved in the cache.

Data flow

â Inputs: active auction snapshots, historical sold data, cached AI Carsales checks, and operator-entered Carsales estimates.

â Outputs: updated ‘ai_listing_valuations.csv‘ plus refreshed bid data when you trigger updates from the page.

Notes

â The page stores manual Carsales inputs per URL in ‘st.session_state‘ so partially entered values persist while you compare vehicles.

6. Missed Opportunities

Purpose

Cross-check recent sale prices against the manual Carsales valuations to highlight deals that should have been bought.

Key capabilities

â¢ Loads cached Carsales tables and ‘sold_cars.csv‘, filters to sold records, and computes the profit gap between the manual average price and the actual hammer price.
â¢ Shows the top three gaps as callouts plus a full table with currency-formatted pricing and odometer stats so you can review the evidence.

Data flow

â Inputs: ‘ai_listing_valuations.csv‘ (for manual Carsales data) and ‘sold_cars.csv‘.
â Output: in-app leaderboard of positive profit deltas for post-mortems.

Notes

â Every calculation normalises the saved text values into numeric form, so even loosely structured Carsales notes can be compared objectively.

7. Outcome Accuracy Tracker

Purpose

Measure how well AI predictions performed once vehicles settled, broken down by time, verdict tier, and individual misses.

Key capabilities

â¢ Requires ‘ai_listing_valuations.csv‘, ‘sold_cars.csv‘, and ‘ai_verdicts.csv‘, then calls ‘scripts.outcome_tracking.compute_outcome_metrics()‘ to assemble joined datasets.

â¢ Displays aggregate KPIs (scored listings, accuracy, MAE, MAPE, profit calibration) plus Altair charts for weekly hit rates and accuracy by verdict tier.

â¢ Provides a detailed Worst Misses table and download buttons for the scored listings, weekly metrics, and verdict metrics CSVs so analysts can dig deeper offline.

Data flow

â Inputs: joined AI verdicts, pricing analyses, and sold outcomes.

â Outputs: exported CSVs for accuracy tracking and on-screen diagnostics.

Notes

â Metrics update inside a spinner whenever the page loads to keep the accuracy dashboard consistent with the freshest data on disk.

99. Style Guide & Template

Purpose

Act as a living design system so new Streamlit pages stay on brand without guesswork.

Key capabilities

â¢ Applies the shared global CSS tokens, displays the AutoSniper colour palette with token names, and demonstrates responsive grid layouts, cards, and button styles.
â¢ Provides copy-ready HTML snippets (banner, palette, button rows) developers can reuse when building future tooling pages.

Notes

â Use this page as a reference when adding UI polish or troubleshooting layout spacing.