iProperty Scraping Notes

URL Extraction Rules

1. React Flight `shareLink`
2. JSON-LD `RealEstateListing.url`
3. `<link rel="canonical">`
4. `<meta property="og:url">`
5. `url_in` (last resort)

Validation Guards

- Domain must be **iproperty.com.my**
- Path matches `^/property/.+/(sale|rent)-\d+/?$`
- Normalize to **https**
- Strip query string and fragment
- Resolve relative canonicals to absolute URLs

Save/Append Policy

- When you say **"save"**, I keep these rules.
- When you add new items, I **append** them here (don't overwrite previous ones).

Future Saved Items

-

Listing ID Extraction Rules

Priority order 1. Flight: `listingDetail.id` 2. Meta table text: e.g., `Listing ID – <digits>` 3. URL-derived: digits from canonical/og/JSON-LD URL path `/property/.../(sale|rent)-<ID>/` 4. Next/State JSON fallback: numeric `"listingId": <digits>`

Validation Guards - ID must be **digits only** (`^[0-9]+$`). - If both Meta and URL yield an ID, they **must match**; if they differ, prefer **Meta table**. - Prefer **Flight id** when multiple candidates exist. - Use **Next/State** only when Meta is missing; if present, cross-check with any URL-parsed ID when the URL follows the standard `/sale|rent-<ID>/` form. - Non-standard paths (e.g., `…/ol-sale-<ID>/`): **do not** parse ID from URL; rely on Meta or Next/State. - Always record the **source** used (`flight | meta | url | next`) for audit/debug. - Output as a **string**, no commas/spaces.

Title Extraction Rules

Short title (e.g., "Aetas Damansara, Tropicana") 1. JSON-LD `RealEstateListing.name` 2. `<h1>` text 3. URL slugs → `Project, Area` (title-cased)

Soft guards (short) - Reject if contains: `for sale|for rent| by | rm | psf | iproperty` (case-insensitive). If rejected, try next source. - Prefer simple shape: 1–2 phrases separated by `,`, `in`, or `at`; length 8–60 chars. - URL fallback: use canonical or og:url on **iproperty.com.my**; take the two slugs before `/sale-…` or `/rent-…`; replace `-`→space and Title-Case. Ignore `ol-sale-*` as a slug. - If JSON-LD/H1 disagree heavily with URL (large edit distance), prefer JSON-LD/H1 and log.

Long title (e.g., "Condominium for Sale in Aetas Damansara, Tropicana") 1. `<meta property="og:title">` 2. `<title>` 3. `<meta name="twitter:title">` (if present)

Soft guards (long) - Must include a transaction cue: `for Sale` or `for Rent`. Otherwise try next source. - Should include a plausible property type from allowlist: {Condominium, Apartment, Serviced Residence, Terrace, Semi-D, Bungalow, Shop-Office, Office, Land, Factory}. If missing, mark suspect and try next source. - Cleanup: strip site branding (`| iProperty.com.my`), agent names (`by …`), REN/licence strings, prices. Normalize spaces/case. - Length target: 20–140 chars after cleanup. Outside → try next source. - Cross-check: long title should generally **contain** the short title; if not, keep short title and flag long as suspect.

Fail-closed behavior - Never hard-fail: if all sources trip guards, keep the first acceptable candidate (post-cleanup). - Log which source was used: `ld | h1 | url | og | title | twitter`.

Date Extraction Rules

Field name: `listing_date` (formerly `latest_date`)

Priority order (listing-scoped only) 1. **Metatable (DOM):** row labeled *"Listed on <date>"* 2. **JSON:** `lastPosted.date` 3. **JSON:** `lastPosted.unix` → convert to Asia/Kuala_Lumpur, take the date (`YYYY-MM-DD`) 4. **JSON/JSON-LD:** `datePosted` or `postedAt` (pick the most recent if both) 5. **JSON-LD:** `datePublished` (last resort)

Quick guards - **Scope guard:** Ignore config/page keys (e.g., `featureToggleConfig.createdAt/updatedAt`). Not listing-scoped. - **Future/ancient guard:** Reject dates in the future (> today) or before year 2000. - **Format/locale:** Prefer DMY (site locale). If numeric ambiguous (e.g., `09/07/2025`), require surrounding cue (e.g., "Listed on") or another source. - **Timezone:** For timestamps (ISO/epoch), convert to **Asia/Kuala_Lumpur**; store as `YYYY-MM-DD`. - **Source tagging:** Save which source was used: `dom:liston | json:lastPosted.date | json:lastPosted.unix | json:datePosted | jsonld:datePublished`.

Notes - Do **not** set `created_date` from config-level fields. If no listing-scoped created date exists, leave `created_date` blank or record an `approx_created_date` separately (earliest of `datePosted`/`postedAt`).

Agent Name Extraction Rules

Priority order 1. `contactAgentData.contactAgentCard.agentInfoProps.agent.name` 2. Flight payload: `listers[0].name` → `lister.agentName` 3. DOM agent link/name: text of `a[href*="/`

`property-agent/"]` or agent card block 4. Page title fallback: extract name from patterns like `… by <Name>`

Guards - Reject placeholders: “Private Advertiser”, agency names (e.g., words like *Realty*, *Sdn Bhd*), or single-word tokens. - Expect 2–4 words, alphabetic-heavy, 3–40 chars; Title-Case normalize. - If multiple candidates exist, prefer the `contactAgentData` value.

Agent ID Extraction Rules

Priority order (most → least reliable) 1. `contactAgentData.contactAgentCard.agentInfoProps.agent.id` (or `agentId`) 2. Flight payload: `listers[0].id` → `lister.agentId` 3. **Profile URL parse:** `/property-agent/<slug>-<ID>` (including `#Agent-RightSideProfile`) 4. JSON agent blocks: `{"agent": {"id": <ID>}}` or standalone `"agentId": <ID>` near contact/agent payloads 5. DOM anchors to `/property-agent/…` (parse trailing `-<ID>`)

Quick guards - Digits only, length **4–9**; **must not equal** `listing_id`. - **Scope check:** ignore IDs under `organisation|organization|agency|user|organisationId` keys. - **Name proximity:** prefer candidates in the same block as the extracted agent **name** (case-insensitive match). - **Tie-breakers:** prefer (1) `contactAgentData` → (2) profile URL → (3) the ID appearing in the most sources. - Record the chosen **source** (`contactAgentData | flight | profileUrl | json.agent | dom.anchor`) for audit.

Property Type Extraction Rules

Priority order (most → least) 1. Flight payload: `listingDetail.propertyType` 2. JSON-LD `RealEstateListing.category` 3. Breadcrumbs: `bc[4]` from `BreadcrumbList` (your current indexing) 4. Metatable “Property details” → extract `<type>` from `"<type> for sale|rent"` 5. **NEW last fallback:** parse from `og:title` / `<title>` → token immediately before `for sale|for rent`

Guards & normalization - Strip any `for sale|for rent` suffix. - Alias map: `Serviced Apartment` ↔️ `Serviced Residence`, `Shop Office` → `Shop-Office`, `Semi D` → `Semi-D`, `Condo` → `Condominium` (extend as needed). - Title-case output; 3–40 chars; alphabetic-heavy. - Record source used: `flight | jsonld | breadcrumb | metatable | headtitle`.

State Extraction Rules

Priority order (most → least) 1. Flight: `listingDetail.languagePlace.level1` (or `multilanguagePlace.enGB.level1`) 2. Breadcrumbs: `bc[2]` from `BreadcrumbList` 3. JSON-LD: `RealEstateListing.spatialCoverage.address.addressRegion` 4. Last-resort (only if 1–3 empty): parse the trailing location token from visible location/meta description **if** it matches the allow-list

Allow-list & normalization - States/FT: {Johor, Kedah, Kelantan, Melaka, Negeri Sembilan, Pahang, Perak, Perlis, **Pulau Pinang↔Penang**, Sabah, Sarawak, Selangor, Terengganu, **Kuala Lumpur**, **Putrajaya**, **Labuan**} - Normalize casing and common variants (e.g., "N. Sembilan" → "Negeri Sembilan", "Penang" → "Pulau Pinang").

Guards - Value **must** be in allow-list; otherwise discard and try next source. - Prefer Flight over Breadcrumb when both exist and differ; if still conflicting, pick the value that also appears in short title/location string. - Do **not** infer state from URL slugs or city names alone. - Record source used: `flight | breadcrumb | jsonld | visible`.

District / Subarea Extraction Rules

Intent: Treat **district** and **subarea** as the same field when only one level is available.

Priority order (most → least) 1. Flight / Next-State place tree: - District: `languagePlace.level2` (or `multilanguagePlace.enGB.level2`) - Subarea: `languagePlace.level3` (or `multilanguagePlace.enGB.level3`) 2. Breadcrumbs: `bc[3]` from `BreadcrumbList` 3. JSON-LD: `RealEstateListing.spatialCoverage.address.addressLocality` 4. URL fallback: from canonical/ og:url **first slug after** `/property/` (Title-Case hyphen-split)

How to populate - If only one of (district, subarea) is found → **copy it** to the other. - Final `location` string = `subarea, district, state` (skip empties).

Guards - Must not equal a known **property type** token (e.g., “Condominium”, “Shop-Office”). - Must not be the **listing_id** or transaction tokens (“sale”, “rent”, “ol-sale-*”). - *Normalize: trim, collapse spaces, Title-Case; 2–40 chars, alphabetic-heavy. - If value equals state** exactly, accept but prefer any non-state alternative from another source. - Record source used: `flight.l2|l3 | breadcrumb | jsonld.locality | url.slug`.

Address Extraction Rules

Priority order (most → least) 1. Flight/Next: `listingDetail.address.formattedAddress` 2. JSON-LD: `RealEstateListing.spatialCoverage.address` → join `{streetAddress, addressLocality, postalCode, addressRegion, addressCountry}` 3. DOM metatable row labeled **Address/Alamat** (value text) 4. Maps link fallback: parse `q` / `query` from `google.com/maps` or `maps.app.goo.gl` anchors → clean 5. Last resort: compose from known parts if ≥2 present (e.g., `{project_or_short_title}, {subarea}, {state}` + `{postalCode}` when available)

Guards & normalization - Strip trailing `, Malaysia` / `, MY`; keep country only if nothing else present. - Prefer values containing a **postal code** (5 digits) or both **locality + state**. - Max length 200 chars; collapse multiple spaces; remove duplicated commas. - Common shortforms (optional): `Jln→Jalan`, `Tmn→Taman` (only if you want normalization; keep original if unsure). - Record source used: `flight.formatted | jsonld.addr | dom.address | maps.link | composed`.

Latitude / Longitude Extraction Rules

Priority order (most → least) 1. Flight/Next: `listingDetail.address.lat` + `listingDetail.address.lng` 2. JSON-LD: `geo.latitude` + `geo.longitude` 3. Meta: `<meta property="place:location:latitude">` + `<meta property="place:location:longitude">` 4. Google Maps anchors: parse `@<lat>,<lng>` from path or `q=` / `query=` parameters

Guards & normalization - Validate ranges: `lat ∈ [-90, 90]`, `lng ∈ [-180, 180]`; non-zero. - Ignore config/tracking blocks; accept only listing-scoped or standard metadata. - Do **not** use generic `"lat": …` proximity heuristics. - Record source used: `flight | jsonld | meta.place | maps`.

Price Extraction Rules

Priority order (most → least) 1. **Flight/Next:** `listingDetail.price.{currency,min,max}` → single `price = min if present else max`; `price_currency = currency` 2. **JSON-LD:** `RealEstateListing.offers.{price, priceCurrency}` 3. **DOM headline (visible):** parse `RM <amount>` from the hero/price area; **ignore** lines containing `psf`, `psm`, `per sq`, `sqft`, `sqm`.
• **Rent mode:** prefer amounts with `/month` or `per month` (or explicit "for rent").
• **Sale mode:** pick the **largest** RM figure among headline candidates. 4. **Head titles (last fallback):** extract from `<title>`, `meta[property="og:title"]`, `meta[name="twitter:title"]` **only** (no descriptions). Require `RM <amount>` and a matching transaction cue (`for sale` / `for rent`).

What to exclude - Do **not** use `meta[name="description"]`, `meta[property="og:description"]`, or `meta[name="twitter:description"]` for price.

Guards & normalization - Currency: map `RM` → `MYR`; ignore other currencies. - Sanity (Malaysia context):
• Sale: accept `price ≥ 10,000`.
• Rent: accept `price ≤ 100,000` and prefer entries with `per month`. - Strip commas, coerce to number; max length for raw strings 32 chars. - Reject any candidate mentioning `psf`, `psm`, `per sq`, or deposit/booking cues unless explicitly in rent headline. - Record source used: `flight.price | jsonld.offers | dom.rm | head.title`.

Built-up / Floor Area Extraction Rules

Priority order (most → least) 1. Flight/Next: `listingDetail.attributes.builtUp` (+ `sizeUnit`) 2. **Next/State structured**: `detailsData.metatable.items` → keyword match in label/value (`built up`, `built-up`, `floor area`, `size`, `builtup`; Malay: `keluasan binaan`, `luas binaan`) → extract first `<number + unit>` 3. DOM metatable (same keyword match as #2) 4. JSON-LD: `RealEstateListing.floorSize` or `additionalProperty` (value+unit)

Normalization - Units accepted: `sq ft | sqft | sf | sq.ft` → **SQUARE_FEET**; `sqm | m² | sq.m | square meter(s)` → **SQUARE_METER**. - Numbers: strip commas; keep 1 decimal if present.

Guards - Ignore values found in floor-plan/gallery captions and long description blocks when a structured/metatable value exists. - Range sanity (residential/office): **400–20,000 sqft** (≈ **37–1,858 sqm**); outside this range requires a second confirming source. - If multiple candidates differ <10%, pick the **smaller**. - Record source used: `flight.attr | next.metatable | dom.metatable | jsonld`.

Built-up PSF (Floor) Extraction Rules

Priority order (most → least) 1. **Flight/Next attributes**: - `listingDetail.attributes.pricePerSizeUnitBuiltUp` (string like `RM 570 psf`) - If missing, use numeric: `attributes.minimumPricePerSizeUnitBuiltUp` → `attributes.maximumPricePerSizeUnitBuiltUp` → `attributes.minimumPricePerSizeUnit` → `attributes.maximumPricePerSizeUnit` (only if clearly **built-up**, see guards) 2. **Next/State computed fields**: numeric `floorAreaPsf` / `builtUpPsf` when present (e.g., `569.95`) 3. **Next/State metatable**: any `metatable.items[*].value` containing `psf` and `(floor)` → extract numeric 4. **DOM metatable**: `.meta-table__item` containing `psf` and `(floor)` → extract numeric 5. **FAQ/QA blocks**: lines like `Current PSF: RM 569.95 psf` → extract numeric (treat as site-provided, not description) 6. **Compute (sale only, last resort):** `price / built_up_sqft` (convert sqm→sqft first). Store as `psf_computed` and keep site-provided PSF as primary when both exist.

Guards & disambiguation - Prefer entries explicitly marked `(floor)` or `Built-up` over generic `psf`. - If both `psf` and `psm` are present, convert `psm → psf` by dividing by `10.7639` and tag source as `psm_converted`. - Sanity ranges: - Sale PSF: `RM 100 – RM 5,000` per sqft. - Rent monthly PSF (if stored): `RM 0.5 – RM 50` per sqft per month. - Normalize to numeric with 2 decimals; currency `MYR`. - Record source used: `attr.psfBuiltUp | attr.min/max | state.floorAreaPsf | state.metatable | dom.metatable | faq | computed`.

Land Size & Land PSF — Safe Fallbacks (no mix-up with Built-up)

Priority (most → least): 1) **Flight/Next attributes**
- `attributes.landArea` (+ `attributes.sizeUnitLandArea`)
- `attributes.pricePerSizeUnitLandArea` → `minimumPricePerSizeUnitLandArea` → `maximumPricePerSizeUnitLandArea` 2) **Next/State metatable (`more-details-widget`)**
- Pair `.amenity-label` with its sibling `.amenity-value`; accept only labels matching: `Land area|Land size|Lot size|Site area|Keluasan tanah|Luas tanah` (case-insensitive) 3) **DOM metatable (same widget)**
- Same label pairing as above. 4) **Scoped hero/Property-details fallback**
- Within the Property details container only; accept values only when the **label** contains the same land keywords above. 5) **Compute `land_psf` (sale only)**
- If `price` present and `land_size` present, convert units to **sq ft** (sqm→sqft, acre→sqft, hectare→sqft) and compute `price/land_sqft`.

Hard Guards (prevent false positives / built-up confusion): - **Require a land label** near the value. Reject any match where the label contains `Built-up|Built up|Floor area|Size (without land)`. - **Scope** to the Property details widget (`dataautomationid="more-details-widget"`). Ignore values from description, galleries, carousels, or related listings. - **Units**: accept only `sq ft|sqft|sf|sqm|m²|sq.m|acre|ac|hectare|ha` for land size. For land PSF, require `psf` on the value **and** a land label; reject generic `psf (floor)`. - **Property type sanity**: if `property_type` ∈ {Condominium, Apartment, Serviced Residence, SOHO, Flat} (strata) **and** no explicit land label/Flight land fields, leave `land_size` and `land_psf` **blank**. - **De-dup with built-up**: if extracted `land_size` equals `built_up` exactly and page is strata, drop `land_size`. - **Ranges** (after unit normalization):

- Land size: 200 sq ft – 10,000,000 sq ft (reject outliers).
- Land PSF: RM 1 – RM 10,000 (reject otherwise). - **Source tagging**: record the source used (`attr.landArea|attr.landPSF|min/max|state.metatable|dom.metatable|hero.details|computed`) for audit.

Title Type — token-only fallback (safe)

When the page lists the value as a standalone item (no label), capture it with strict scope and a whitelist of exact tokens.

Priority insert (between metatable-labeled and DOM-labeled steps): - 2b) Next/State metatable (token-only value) - Path: detailsData.metatable.items[*].value - Accept only if the value text equals one of: "Strata title", "Individual title", "Master title" (case-insensitive, trimmed). - 3b) DOM metatable (token-only value) - Scope: [dataautomationid="more-details-widget"] .meta-table__item .amenity-value (or .meta-table__value) - Same whitelist as above.

Guards (to avoid false positives): - Scope strictly to the Property details widget. Ignore description/FAQ/sidebar/related areas. - Do not match generic words like "title"; this branch is value-only. - Exclude tenure tokens ("Freehold", "Leasehold") and any line containing "psf", "floor", or "built-up". - Ignore meta tags like og:title, twitter:title, and modal label elements (...-body-title). - Normalize outputs to: Strata | Individual | Master.

Source tagging: state.metatable.token, dom.metatable.token. No inference: store blank if no whitelist hit.

Bumi Lot — extraction & fallbacks (keep full text)

Goal: Store both the raw phrase (e.g., `"Not Bumi Lot"`) and a normalized boolean.

Priority (most → least): 1) **Next/State metatable (token-only value)**
- Path: `detailsData.metatable.items[*].value`
- Accept only when the value contains the token `Bumi` (case-insensitive).
- Save `bumi_lot_text` = exact string; derive `is_bumi_lot` via rules below. 2) **DOM metatable (token-only value)**
- Scope: `[dataautomationid="more-details-widget"] .meta-table__item .amenity-value` (or `.meta-table__value`)
- Same `Bumi` token check and derivation. 3) **Flight boolean (fallback when no text is available)**
- `listingDetail.attributes.isBumiLot` → set `is_bumi_lot` only.
- Leave `bumi_lot_text` **blank** (do **not** synthesize text).

Normalization (boolean derivation from text): - Let `t = lower(bumi_lot_text)`. - **False** if `t` contains any negation: `not`, `non`, `non-bumi`, `non bumi`, `bukan`, `bkn`, `no` (e.g., `"Not Bumi Lot"`, `"Non-Bumi Lot"` → `is_bumi_lot = False`). - **True** if `t` contains `"Bumi"` and no negation tokens (e.g., `"Bumi Lot"`).

Guards (avoid false positives): - **Scope strictly** to Property details widget; ignore description/FAQ/sidebar/ related sections.
- Require the `Bumi` token in the value; do **not** match generic lines (e.g., `Title type`, `Freehold tenure`).
- Do **not** infer from property type or location.
- If both metatable text and Flight boolean are present but disagree, **prefer the metatable text** (surface text is authoritative for the snapshot) and record **source tag**.

Source tagging: `state.metatable.token`, `dom.metatable.token`, `flight.attributes.isBumiLot`.