PropertyScout

The Complete AI-Powered Property Investment Platform

White Paper

August 11, 2025

Version 1.0 — Confidential Draft for Early Partners and Reviewers

Contents

Executive Summary

PropertyScout is an AI-first property investment platform that automates the end-to-end workflow of sourcing, underwriting, and monitoring UK residential deals. The system integrates automated scraping, robust data validation, AI-based refurbishment cost estimation from listing photos, rental forecasting, cashflow and ROI modeling, EPC matching, forecasting of value uplift, and report generation (PDF and Excel). It is designed for investors, sourcing agents, and estate agencies seeking faster, more consistent decisions with auditability and repeatability.

This document details the current MVP, planned features, reference architecture, data models, algorithms, evaluation frameworks, regulatory controls, commercialization strategy, and a structured roadmap. The goal is to provide technical and commercial clarity for pilot customers, design partners, and potential investors.

Positioning. Unlike calculators or point tools, PropertyScout focuses on (i) automation across upstream data capture, (ii) AI-driven analysis with human-in-the-loop validation, and (iii) post-purchase monitoring and alerts—creating a continuous value loop.

Outcome. Reduced analysis time from hours to minutes; consistent underwriting and documentation; proactive alerts; and the ability to scale to hundreds of deals per analyst per week.

Problem and Opportunity

Industry Pain Points

- **Fragmented tooling:** Data sourcing, underwriting, EPC checks, and reporting often require multiple systems and manual reconciliation.
- **Latency and inconsistency:** Human analysis varies widely; many deals are lost due to slow turnarounds or inconsistent assumptions.
- **Lack of monitoring:** After purchase, few tools track remortgage windows, interest-rate impacts, or portfolio KPIs in real time.
- **Regional nuance:** Refurb costs and rents are hyper-local; generic calculators miss local reality.

Why Now

- **Mature LLMs and vision models** enable robust extraction and estimation from messy sources (HTML, images).
- **API growth** (EPC, ONS, Land Registry, portal data) improves coverage and freshness.
- **Investor demand** for faster, defensible underwriting and automated sourcing/alerts is rising as yields compress.

Product Overview

Elevator Pitch

PropertyScout is the ultimate AI-powered investment assistant that ingests a property link, extracts structured data, estimates refurb costs from photos, forecasts rents and values, computes full-stack financials, attaches EPC data, and generates client-ready reports—then continues to monitor and alert post-purchase.

Current MVP Capabilities

1. URL input → **scrape & parse** (price, agent, layout, media, key features).
2. **AI refurbishment estimation** from images with room-level breakdowns.
3. **Rental estimation** using GPT with postcode comparables.
4. **Financial stack:** stamp duty, legal/survey/insurance, management fees, yield & ROI.
5. **Value uplift** via HPI-baseline + AI adjustment.
6. **EPC matching** to retrieve rating and details.
7. **Reporting:** PDF and Excel export; Slack notification with attachments.

Planned Features (Highlights)

- Live portal integrations (Rightmove, Zoopla, auctions) with filters, de-dup, and watch-lists.
- User-configurable alerting on yields, price drops, new listings.
- Post-purchase mortgage optimisation, remortgage timing alerts.
- Portfolio dashboard, deal comparisons, and client-shareable views.
- Off-market CRM tools (compliant outreach) and letter generation.

User Personas & Jobs-To-Be-Done

Professional Investor

JTBD: "Underwrite deals consistently and fast; produce investor-ready packs; monitor portfolio KPIs and remortgage opportunities."

Sourcing Agent

JTBD: "Scan and qualify many deals daily; package them for buyers with credible numbers; run targeted alerts."

Estate Agency (Partner)

JTBD: "Provide value-add analytics to buyers; white-label reporting; convert more instructions; evidence-based pricing."

System Architecture

High-Level Diagram

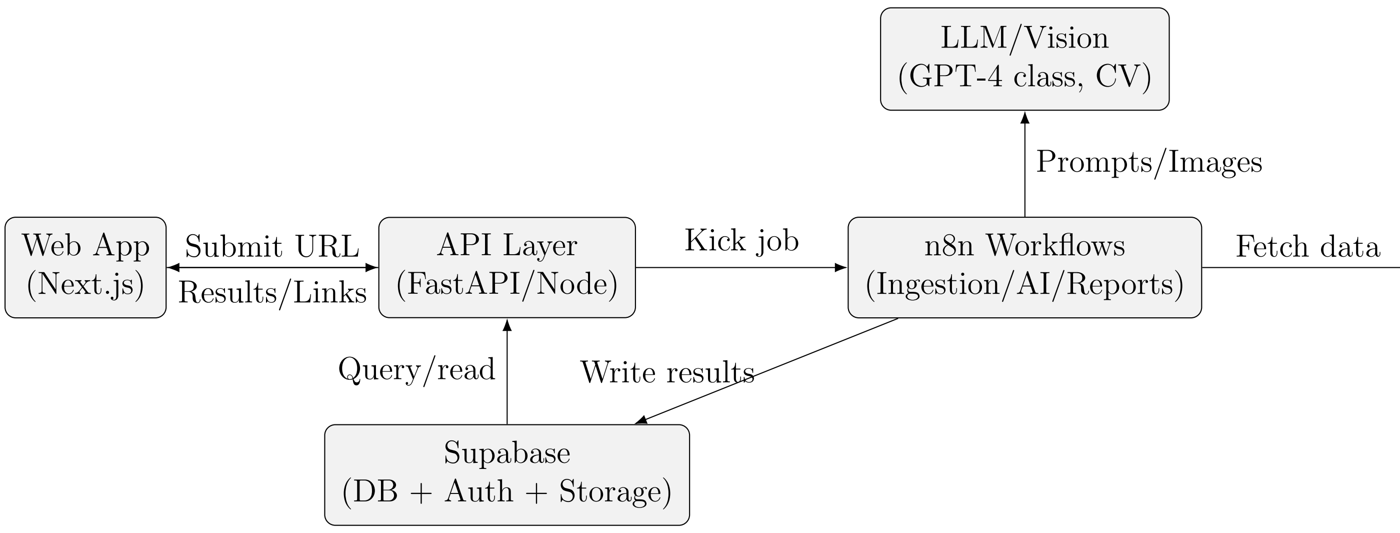

Figure 1: PropertyScout high-level architecture.

Workflow Orchestration (n8n)

1. **Scrape Node:** Fetch HTML and media links.
2. **Clean & Parse Agent:** Convert HTML to structured JSON (price, address, features).
3. **Image Loop:** For each image, run a refurb-estimate prompt, aggregate per room.
4. **Rent Agent:** Estimate rent using postcode/regression prior + GPT adjustment.
5. **Fees & ROI Node:** Compute stamp duty, legal, management, net yield, ROI, cashflow.
6. **HPI & Uplift Node:** Calculate baseline uplift; optional AI adjustment.
7. **EPC Match Agent:** Find EPC record and extract rating + fields.
8. **PDF & Excel Nodes:** Generate reports; upload to storage.
9. **Notify:** Slack/email with links and summary metrics.

Data Stores

Supabase: Postgres for canonical tables (properties, analyses, comps, epc, runs), Storage for PDF/Excel artifacts, Auth for users/teams.

Data Model (Canonical Schemas)

Core Entities

Table	Purpose & Key Fields
`properties`	Canonical property identity. *Key fields:* id (uuid), url, address_full, postcode, source, first_seen, last_seen, status.
`analyses`	One analysis per run/version. *Key fields:* id, property_id (fk), run_timestamp, price, sq_ft, bedrooms, bathrooms, assumptions jsonb, results jsonb.
`refurb_estimates`	Room- or item-level refurb costs. *Key fields:* id, analysis_id (fk), room_type, items jsonb, subtotal, confidence.
`rent_estimates`	Rent predictions and sources. *Key fields:* id, analysis_id (fk), monthly_rent, method, comps jsonb, confidence.
`fees`	Stamp duty, legal, survey, insurance, mgmt. *Key fields:* id, analysis_id (fk), duty, legal, survey, insurance, mgmt_pct, total.
`sales_comps`	Nearby sales comparables. *Key fields:* id, analysis_id (fk), address, distance_m, price, date, sqft, ppsf.
`rental_comps`	Nearby rental comparables. Fields similar to sales comps.
`epc_records`	EPC details linked to property. *Key fields:* id, analysis_id, rating, uprn, lmk_key, measures jsonb, full_record jsonb.
`reports`	Generated artifacts. *Key fields:* id, analysis_id, type (pdf/xlsx), storage_path, size_bytes, checksum.
`runs`	Orchestration metadata. *Key fields:* id, user_id, workflow_id, duration_ms, status, error_log.

Algorithms & Methods

HTML → JSON Extraction

Method: Heuristic tag selection + LLM cleanup. We use deterministic rules for price/address hints, and an LLM to standardise fields and fill light gaps. Ambiguous fields produce warnings rather than hallucinated values.

Vision-Guided Refurb Estimation

Input: Listing photos; optional captions.
Process:

1. Image classification: detect room type (kitchen, bathroom, bedroom, lounge, hallway, exterior).
2. Condition assessment: extract cues (e.g., old cabinets, tile condition, damp marks).
3. Bill-of-quantities template: map detected items to line-items (e.g., "replace cabinets", "rewire room").
4. Regional cost adjustment: apply postcode-based cost indices.

5. Confidence scoring: reflect image quality, occlusion, and ambiguity.

EPC Matching (Robust Heuristic)

Goal: From candidate EPC records, select the best match for a given `propertyTitle` and `postcode`.
Steps:

1. Normalise: uppercase; remove punctuation; collapse whitespace.
2. Extract tokens: house number & primary street tokens.
3. Filter by postcode equality (normalised).
4. Score each record:

   $$s = w_h \cdot \mathbb{1}[\text{house\# match}] + w_s \cdot \text{Jaccard}(\textit{street tokens}) - w_e \cdot \text{editDist}(\textit{title}, \textit{address})$$

5. Choose max-s; threshold; else null.

Weights w_h, w_s, w_e are tuned from labeled pairs. This approach is fast (linear in candidates) and avoids LLM latency.

Rent Estimation

Hybrid model: Baseline regression using postcode + property features; GPT layer adjusts based on unstructured signals (listing language, amenities, finish quality). Final:

$$\hat{R} = \alpha \cdot R_{\text{regression}} + (1 - \alpha) \cdot R_{\text{GPT}}$$

with α calibrated by backtesting per region.

Fees & ROI

Given purchase price P, refurb cost C_r, other fees F, annual rent R_a, operating expense ratio e, and management fee μ:

$$\text{NOI} = R_a \cdot (1 - e - \mu) \qquad \text{Yield} = \frac{\text{NOI}}{P + C_r + F}$$

For leverage, monthly mortgage M via standard amortisation; monthly cashflow $\text{CF}_m = \frac{\text{NOI}}{12} - M$.

Uplift & Forecasting

Baseline: Apply HPI growth to P; **Post-refurb uplift** Δ from comps or AI adjustment; forecast price P':

$$P' = (P + C_r) \cdot (1 + g_{\text{HPI}}) + \Delta$$

Sensitivity analysis runs scenarios over g_{HPI}, R_a, and C_r with tornado chart outputs (in Excel/PDF).

Prompts & Guardrails (Selected)

HTML-to-JSON Prompt (Skeleton)

```
System: You are a strict data extractor. If a field is unknown,
   return null.
User: Given raw HTML and images, return ONLY valid JSON with keys
   :
{price_gbp, address, postcode, bedrooms, bathrooms, tenure, agent
   , features[]}
Do not invent values. Use best-guess only if explicitly stated by
    the page.
```

Refurb Estimation Prompt (Vision)

```
System: You estimate refurbishment line-items from photos, using
   UK trade assumptions.
User: For each image, classify room type, list line-items with
   qty, unit_cost_gbp, subtotal,
and a confidence 0-1. Adjust costs for postcode if provided.
   Output valid JSON only.
```

Rent Estimation Prompt

```
System: You estimate monthly rent in GBP for UK properties.
User: Using the provided structured features and any text cues,
   output:
{ monthly_rent_gbp, low, high, rationale, confidence }.
Don't exceed local comparables by >15% without specific
   justification.
```

EPC Matching Prompt (LLM fallback)

```
System: You match a single EPC record to a title+postcode and
   return exactly one JSON object.
User: Inputs: propertyTitle, postcode, epcRecords[]. If unsure,
   return nulls (no guess).
No markdown. Only JSON.
```

Evaluation & Accuracy Targets

Metrics

- **Rent MAE / MAPE:** $|\hat{R} - R|$, percentage error vs. let-agreed rents.

- **Refurb Cost Error:** absolute/relative error vs. post-job invoices or QS estimates.
- **EPC Match Precision/Recall:** on a labeled set.
- **Overall Deal Error:** % error in Year-1 net yield and cashflow.

Initial Targets (Beta)

- Rent MAPE: 10–15% (postcode & 1–2 bed flats), 15–20% (mixed stock).
- Refurb error: 20–30% at room-bundle level; 15–25% with ¿10 images.
- EPC matching: >95% precision on clean addresses; >90% on messy.
- Year-1 net yield error: ±1–1.5pp for standard BTL scenarios.

Test Protocol

1. Build hold-out set by region and property type.
2. Backtest on historical listings with known outcomes (rents, invoices).
3. Report aggregate and per-region metrics; track drift monthly.

Security, Compliance, and Data Governance

GDPR

- Lawful basis (Legitimate Interests/Contract) for processing listing data.
- Data minimisation: only persist fields needed for analysis/audit.
- User consent flows for outreach features (opt-in).

Credentials

All secrets stored as environment variables or secret managers. Keys redacted in logs and reports. **Never embed API keys in code or documents.**

PII and Retention

- Default retention: analysis artifacts retained 12 months (configurable).
- Right to erasure pipeline; access logging; DPA-ready documentation.

Performance & Cost

Latency Budgets (MVP)

- End-to-end target: 30–90 seconds per property (depends on image count).
- Sub-budgets: HTML parse < 3s; images loop < 40s (10–15 images); EPC/API calls < 5s; PDF/Excel < 7s.

Run Cost (Illustrative)

- LLM tokens (parse + images + rents + EPC): £0.05–£0.20 per run (model/size dependent).
- Storage (PDF/Excel): pennies per artifact.
- Overall COGS per run (MVP): £0.10–£0.40 typical.

Commercial Strategy

Pricing

- **Starter:** £29/mo, 50 runs, pay-as-you-go at £1/run thereafter.
- **Pro:** £99/mo, 300 runs, team seats, alerts, PDF/Excel branding.
- **Team/Agency:** £299+/mo, 1,000 runs, SSO, white-label, priority support.
- **Custom Region Builds:** One-off setup & premium monthly (e.g., North West focus), higher accuracy guarantees in coverage zone.

GTM

- Beta cohort (10–20 users) with discounted lifetime tiers.
- Partnerships with sourcing groups & agents; co-marketed case studies.
- Thought leadership: white papers, accuracy dashboards, regional insights.

Unit Economics (Indicative)

- Gross margin target: >80% at scale with caching and heuristics replacing some LLM calls.
- Key levers: model selection, prompt efficiency, deduped runs, batch scoring, regional model fine-tunes.

Risk Register & Mitigations

Risk	Description	Mitigation
Data drift	Rents/refurb costs shift regionally.	Monthly backtests; retrain priors; add user feedback weighting.
API fragility	Portal/third-party API changes.	Abstraction layer; fallback scrapers; monitoring & alerts.

LLM variability	Different outputs/version changes.	Strict JSON schema validation; retries; guardrails; caching.
Compliance	Outreach & data usage.	GDPR DPIA; opt-in flows; audit logs; updated DPA.
COGS creep	Token costs rise.	Smaller models for easy tasks; heuristic prefilters; batch calls.

Roadmap

6-Week Plan (Detailed)

Week	Objectives
1	n8n + Supabase setup; scraper hardened; HTML→JSON prompt; baseline schemas.
2	Basic dashboard; rent regression prior; parameter toggles; initial Excel layout.
3	E2E tests; Slack/email alerting; PDF polish; EPC matching hardening.
4	Off-market CRM MVP; probate/distressed alerts; basic outreach templates.
5	Post-purchase dashboard (rates feed stub); remortgage alert logic.
6	Stress test; docs; demo kit; beta onboarding; support playbooks.

Post-MVP (Quarterly Themes)

- **Q1:** Interest-rate forward curve; prompt-tuning pipeline; API quotas & scaling.
- **Q2:** Regional fine-tunes; deeper comps; A/B of hybrid rent models.
- **Q3:** White-label; embeddable widgets; team workflows.
- **Q4:** Advanced portfolio analytics; acquisition-readiness hardening.

Case Study (Illustrative)

Deal Summary

- Guide price: £525k; 2-bed flat; W2 postcode.
- Refurb from images: kitchen refresh, bathroom mid-grade, redecoration.
- Rent estimate: £2,650/mo (low £2,450, high £2,850).

Outputs

- Year-1 net yield: 4.9% (no leverage); ROI (with leverage): 6.8%.
- Uplift post-refurb: +6.5 % baseline; EPC: C (recommended measures: glazing).
- PDF generated with breakdown; Slack alert to buyer and agent.

Support, SLAs, and Operations

Beta Support

- Email/Slack within 24h; incident comms; weekly build notes.
- Feedback capture in-product; accuracy scorecards per user.

SLA Targets (Post-Beta)

- 99.5% uptime monthly; P1 incidents < 4h to mitigate; P2 < 1 business day.

Conclusion

PropertyScout delivers a practical, automation-first approach to deal analysis: fast ingestion, structured outputs, transparent assumptions, and continuous monitoring. The roadmap emphasizes accuracy, compliance, and operational reliability. With disciplined execution, the platform can become a category-defining operating system for UK residential investors and partners.

Appendix A: Supabase SQL DDL (Selected)

```sql
-- Note: replace schema and tighten types/constraints per
   environment.
create table public.properties (
  id uuid primary key default gen_random_uuid(),
  url text not null,
  address_full text,
  postcode text,
  source text,
  first_seen timestamptz default now(),
  last_seen timestamptz default now(),
  status text default 'active'
);

create table public.analyses (
  id uuid primary key default gen_random_uuid(),
  property_id uuid references public.properties(id) on delete
     cascade,
  run_timestamp timestamptz default now(),
  price numeric,
  sq_ft numeric,
  bedrooms int,
  bathrooms int,
  assumptions jsonb not null default '{}',
  results jsonb not null default '{}'
);

create table public.refurb_estimates (
  id uuid primary key default gen_random_uuid(),
  analysis_id uuid references public.analyses(id) on delete
     cascade,
  room_type text,
  items jsonb not null,
  subtotal numeric,
  confidence numeric
);

create table public.rent_estimates (
  id uuid primary key default gen_random_uuid(),
  analysis_id uuid references public.analyses(id) on delete
     cascade,
  monthly_rent numeric,
  method text,
  comps jsonb,
  confidence numeric
);

create table public.epc_records (
  id uuid primary key default gen_random_uuid(),
  analysis_id uuid references public.analyses(id) on delete
     cascade,
  rating text,
  uprn text,
  lmk_key text,
  measures jsonb,
  full_record jsonb
);

create table public.reports (
  id uuid primary key default gen_random_uuid(),
  analysis_id uuid references public.analyses(id) on delete
     cascade,
  type text check (type in ('pdf','xlsx')),
  storage_path text not null,
  size_bytes bigint,
  checksum text,
  created_at timestamptz default now()
);
```

Appendix B: n8n Workflow (Pseudo-YAML)

```
nodes:
  - Webhook: trigger { url, userId }
  - HTTP: fetch HTML
  - Agent: clean HTML -> JSON (price, address, features)
  - Loop: images[]
      - Agent: refurb estimate per image
  - Code: aggregate refurb lines -> totals
  - Agent: rent estimate (postcodes + comps)
  - Code: fees, stamp duty, management, ROI
  - HTTP: HPI fetch -> uplift baseline
  - HTTP: EPC API -> candidate records
  - Agent: EPC match -> rating + record
  - PDF: compose report
  - Excel: compose spreadsheet
  - Supabase: upsert all tables
  - Slack: send summary + links
```

Appendix C: Report Structure (PDF/Excel)

PDF Sections: Cover → Summary KPIs → Financials → Refurb lines → Rent & comps → EPC → Forecasts → Disclaimers.
Excel Tabs: Summary, Inputs, Refurb, Rent, Fees, EPC, Forecasts, Comps, Audit.

Appendix D: API Stubs (FastAPI)

```python
from fastapi import FastAPI, HTTPException
from pydantic import BaseModel
import uuid

app = FastAPI()

class RunRequest(BaseModel):
    url: str
    user_id: str

class RunResponse(BaseModel):
    job_id: str
    status: str

@app.post("/analyze", response_model=RunResponse)
def analyze(req: RunRequest):
    job_id = str(uuid.uuid4())
    # enqueue n8n webhook here
    return RunResponse(job_id=job_id, status="queued")
```

Appendix E: Prompt Library (Copy-Paste Ready)

Strict JSON Output Guard

```
System: Output only VALID JSON. No markdown, no comments. If
   unsure, use null.
If any schema key is missing, add it with null.
```

EPC Match (Final)

```
System: You are a deterministic matcher. Do not guess.
User: Inputs: { "propertyTitle": "...", "postcode": "...", "
   epcRecords": [...] }
Rules: normalise strings; exact postcode match required; penalise
    house-number mismatch;
if none cross a confidence threshold, return matched_epc: null,
   epc_rating: null.
Output schema:
{
  "property_title": "...",
  "postcode": "...",
  "matched_epc": { ... } | null,
  "epc_rating": "A"|"B"|...|null
}
```

Appendix F: Glossary

- **EPC:** Energy Performance Certificate.
- **HPI:** House Price Index.
- **NOI:** Net Operating Income.
- **MAPE:** Mean Absolute Percentage Error.