This is a technical map of everything in `streamlit_app.py`: every chart, table, metric, and calculation. It covers (1) how each thing is built, (2) how to read it, and (3) what to do with it. Tone is intentionally engineering-first and grounded.

---

1) Data plumbing, config, and guards

Data loading & caching

- **What it does:** `load_df(None)` pulls `cleaned_large_bgg_dataset.parquet` (preferred) or `cleaned_large_bgg_dataset.csv` from disk. `@st.cache_data(ttl=3600)` keeps it hot for an hour.
- **How it works:** Tries parquet → csv, otherwise halts with an error. All exceptions are surfaced.
- **Why it matters:** Eliminates UI state jitter and prevents slow reloads mid-analysis.
- **Read it like this:** If you see “No dataset found…”, the app never reached analytics; fix paths first.

Column partitioning

- **What it does:** `split_features(df)` separates modelable numeric features (`X`) from metadata (`meta`) and removes excluded fields (`EXCLUDE_FOR_CLUSTERING`).
- **How it works:** Keeps numeric columns, fills NA with 0, preserves `Name` in metadata.
- **Why it matters:** Prevents leakage (ratings, rank, owners) into clustering and keeps numeric shape stable.

Global constants & theming

- Color palettes (`PALETTE`, `CHART_COLORS`) and CSS are set early to keep visuals consistent and readable (light backgrounds, muted text, clear hover states).

---

2) Sidebar controls (the “contract” for every downstream view)

Why this matters

- All charts/tables are **post-filter**. If a chart looks empty or odd, check the filter state first.
- **Mechanic/theme matching pitfall:** “All” can starve the result set fast; use “Any” when exploring.

---

3) Clustering & embedding (market segmentation)

K-means + PCA scaffold

- **What it does:** `fit_clusterer(X_all, k)` standardizes numeric features, chooses `k_eff` safely (handles low-var cases), runs K-means (n_init=10), and produces 2-D coordinates with PCA for plotting.
- **How to read:** Clusters are **behavioral/structural** segments across mechanics, players, time, etc. They’re not “themes”; they’re adjacent neighborhoods in feature space.
- **Labeling:** `generate_cluster_labels(view_f)` names clusters using percentiles of complexity/playtime, mean rating (“weakly/solid/strongly”), and **over-indexed** mechanics/themes (`_top_diff_features`). If insufficient signal, falls back to “Segment X: similar to <best example>”.

What to do with it

- Use segment labels to anchor discussion: *“Mid-weight, strongly rated, with worker-placement bias (medium play)”* is more precise than “like mid Euros.”
- If a label looks wrong, it’s a data artifact; drill into the top mechanics/themes list inside Segment Explorer (below).

---

4) Derived fields & guardrails

- **`Play Time Hours`** = minutes/60, clipped to 10h (for axes sanity).
- **`Success Score`** = `(AvgRating − 5) * Owned Users / 1000` — quick, transparent signal combining quality & reach.
- **`Market Penetration`** = `Owned Users / max(Owned Users)` — normalizes reach to [0,1].

Interpretation: `Success Score` is not a profit estimate. It’s a coarse “did this travel?” indicator relative to rating.

---

5) Home KPIs (top metrics row)

- **Total Games (filtered)** and new this year (delta).
- **Market Segments:** count of unique cluster labels in view.
- **Median Rating / Median Owners / Avg Complexity / Success Rate (≥7.0).**

How to read: These are **context baselines** for all subsequent charts. If median rating is 6.2, don’t set a 7.8 target without a differentiator strategy.

---

7) Design Wizard tab

7.1 Archetypes (presets + grounded insights)

- **What it does:** 3×3 chooser with defaults for cats/mechs/weight/time/players and a **market insight** that’s terse, numeric, and specific.
- **How to use:** Pick the closest lane, then customize in the form.

7.2 Design form → Profile → Segmenting → Neighbors

- **Mapping:** Mechanics/themes toggles map to the dataset’s one-hot columns (with guard for “Cooperative Game” and “Solo Mode”).
- **Cluster assignment:** Input is aligned to `X_all`, scaled with the clustering scaler, and assigned a `cluster_id`.
- **Nearest neighbors:** Within that cluster, we compute pairwise distances in standardized space and take the `topn` closest.

Interpretation: Your “similar games” are **feature-space** nearest neighbors, not semantic clones. They are the right baseline for predictions and pricing.

7.3 Predictions (rating, owners, risk)

- **Models:** Tries to load `rating_xgb` and `sales_xgb`. If missing, falls back to **neighbor averages** with small noise for rating and median for owners.
- **Confidence:** Based on neighbor distance dispersion (tighter = higher). Clipped [40,95] to avoid false precision.
- **Percentile:** Where your predicted rating sits in the filtered market.

How to read:

- **Predicted Rating:** Directional. If it’s high but owners low → likely niche; if owners high but rating mid → fun toy, tighten decisions.
- **Confidence:** A proxy for **model agreement** given your feature profile; low confidence → explore a second preset or adjust features (players/time) and recompute.

7.4 Pricing & Unit Economics

Inputs: MSRP, channel fees, unit COGS, shipping, returns, fixed costs, elasticity, sales window.

Core formulas:

- Anchor price (heuristic):

`base = 35 + (complexity − 2.5)*6 + component/premium adders + players/playtime adders → clipped [15,150]`

- Net per unit: `msrp * (1 − channel_fee_pct)`
- Gross profit/unit: `net_per_unit − (unit_cogs + shipping)`
- Owners adjustment (elasticity, optional): `owners_adj = owners_base * (msrp/anchor_price)^elasticity` (clipped to ±40%)
- Effective units: `owners_adj * (1 − returns_pct)`
- Total GP: `gross_profit_per_unit * max(effective_units, 0)`
- Net profit: `Total GP − fixed_costs`
- ROI multiple: `Net profit / fixed_costs` (∞ if fixed=0)
- Break-even units: `fixed_costs / gross_profit_per_unit`
- Payback months: `ceil(breakeven_units / (effective_units / sales_window))`
- Gross margin % (unit): `gross_profit_per_unit / net_per_unit`

9) Segment Explorer tab

selected segment.

- **What to do:** Prototype the top 1–2 changes; don’t stack five.

---

10) Mechanic Synergies tab

- **How to use:** If you've already pinned your complexity target, this narrows pairing options fast.

---

11) Narrative system

- **What it does:** `narr(md)` blockquotes explanatory text when "Show narrative insights" is on (sidebar).
- **Why it matters:** Keeps prescriptive copy out of code paths and lets you toggle it for focused data views (e.g., demos vs deep work).

---

12) Model and input alignment (robustness)

Aligning inputs to training (`align_profile_to_training`)

- **Type coercion:** Numeric cols coerced; bad inputs fall back to 0 (harsh but safe).
- **Consistency:** If `Min Players > Max Players`, we correct.
- **One-hot groups:** Ensures at least one `Cat:` or `Mechanic_` feature is on if a group is present (prevents all-zeros traps).
- **Clamping:** If an input scaler is available, numeric inputs are clipped to the model's training min/max.

Why it matters: Keeps predictions from undefined territory and prevents "nans exploding the tree."

---

13) Performance & UX

- **Caching:** Data (`@st.cache_data`) and models (`@st.cache_resource`) minimize compute churn.
- **Plot ergonomics:** Unified hover modes where helpful; colorbars added explicitly when Plotly won't auto-explain encodings.
- **Legend gaps:** The mechanics network builds a manual legend (dummy traces) because every edge is its own trace.

---

14) How to interpret the entire console (meta-guidance)

- **Don't treat descriptive trends as causal.** Use them to set **bounds** and to

15) Known edge cases / failure modes

- **Empty views:** Over-tight filters → “No games match…” Fix filters first.
- **Skewed predictions:** If neighbors are sparse (tiny segment), confidence will drop. Use a nearby preset to triangulate.
- **Elasticity sensitivity:** If ROI flips sign with ±$5 price changes, you’re bandwidth-limited by assumptions (fees, returns, or COGS). Stress-test those.
- **Mechanic columns mismatch:** If your dataset uses different prefixes or spellings, discovery helpers (`discover_mechanics`, `discover_themes`) will include them but presets may not light them up—toggle manually.

16) Quick index by artifact (what it is → where it lives → how to read

- **Pairings by complexity** → `tab_synergies` → pick pairs matched to your target weight.

If you want this converted to a printable PDF handout or a markdown doc bundled with screenshots and equations, say the word and I'll export a clean version organized for a paper (TOC, figure callouts, and an appendix with formulas).