# 🔧 Real Estate Evaluator: Fixes & Future Enhancements (Sept 29)

### Need to also add: In commercial real estate, you often see IRR, equity multiple, or annualized ROI used as comparative metrics, rather than final-year ROI alone. Confirmed Next Features for Engineering

These are the **next 3 features** we’ll be engineering across your Python files (main.py, calculations.py, pdf\_generator.py):

| **Feature** | **Purpose** | **Integration Target** |
| --- | --- | --- |
| **1. Manual CapEx Inputs** | Model rehab costs + value boosts | main.py (UI), calculations.py, pdf\_generator.py |
| **2. Depreciation Tracker** | Show tax impact of straight-line depreciation | calculations.py, pdf\_generator.py |
| **3. Plotly Graphs with Hover** | Interactive graphing (ROI, Cash Flow, Rent) | Replace matplotlib in main.py (or real\_estate\_eval\_git.py) |

### Planning What Needs to Change

#### 1. Manual CapEx Inputs

You’ll need to add a **form section** in the Streamlit UI to collect entries like:

| **Field** | **Example** |
| --- | --- |
| Description | Roof Repair |
| Amount | $12,000 |
| Year Applied | Year 2 |
| Rent Boost | +$75/month |
| Appreciation Impact | +$6,000 |

**Changes required:**

* **main.py** or real\_estate\_eval\_git.py: Add a CapEx entry form with these fields.
* **calculations.py**: Deduct the CapEx cost from the appropriate year’s cash flow, and boost rent or final value accordingly.
* **pdf\_generator.py**: Display CapEx data in a new CapEx table section in the PDF.

#### 2. Depreciation Tracker

You’ll implement straight-line depreciation for residential (27.5 years) or commercial (39 years).  
depreciation = purchase\_price / 27.5  
  
**Changes required:**

* **calculations.py**: Create calculate\_depreciation() returning yearly and cumulative values.
* **pdf\_generator.py**: Add a small table summarizing depreciation per year.
* **Optional UI**: Add radio select: “Residential (27.5)” or “Commercial (39)”

#### 3. Plotly Interactive Graphs

Replace your existing matplotlib chart with Plotly and support:

| **Action** | **Result** |
| --- | --- |
| Hover | See ROI, Rent, Cash Flow per year |
| Zoom/Pan | Focus on year ranges |
| Export | Optionally export to CSV or PNG |

**Changes required:**

* **main.py**: Import Plotly (plotly.graph\_objects as go) and build traces:

go.Figure([

go.Scatter(x=years, y=roi\_values, name='ROI', hoverinfo='x+y'),

go.Scatter(x=years, y=rent\_values, name='Rent'),

go.Scatter(x=years, y=cashflow\_values, name='Cash Flow')

### ]) 📄 Document Update Summary

The latest Word document already includes:

* Feature roadmap table ✅
* Sample CapEx entry table ✅
* Depreciation mock table ✅
* Plotly hover explanation ✅

**Let me know if you want a PDF version or a separate README.md draft for GitHub including this roadmap.**

### 🤔 So When Does Average ROI Make Sense?

Average (or annualized) ROI is more helpful:

* **When the horizon is uncertain** or open-ended
* When comparing **multiple deals of varying durations**
* For **benchmarking** across platforms (e.g. Fundrise, RealtyMogul) that advertise *average annual returns*
* In scenarios where **returns fluctuate sharply** over years (e.g. a deal front-loaded with losses but back-loaded with gains)

✅ 1. Final Year ROI Labeling — Renamed 'ROI (%)' to 'Final Year ROI (%)' in both PDF and UI.

✅ 2. Annual Cash Flow = First Year Cash Flow — Verified and labeled clearly.

✅ 3. Mortgage Validation — Monthly Mortgage ($) calculations (e.g., $2035.10) verified against inputs.

✅ 4. Duplicate Row Fix — Removed '10yr Cash Flow' and '10Yr Cash Flow' from PDF to avoid repetition.

✅ 5. Metric Renaming — Renamed 'roi\_list' to 'Annual ROI % (by year)' and '10yr Rents' to 'Annual Rents $ (by year)'.

### ✅ 6. Rounding Standardization — Applied consistent rounding across all monetary and percentage values. Here is the Rounding Rules Table we finalized 3 days ago:

| **Metric Type** | **Rounding Rule** | **Example** |
| --- | --- | --- |
| ROI (%) | Round to **2 decimals** | 224.19384 → 224.19 |
| Cap Rate (%) | Round to **1 decimal** | 5.987 → 6.0 |
| Cash-on-Cash Return (%) | Round to **2 decimals** | -0.34327 → -0.34 |
| Monthly Mortgage ($) | Round to **2 decimals** | 1516.9638 → 1516.96 |
| First Year Cash Flow ($) | Round to **2 decimals** | -203.563 → -203.56 |
| Multi-Year Cash Flow (list) | Round each to **2 decimals** | [123.456, 789.123] → [123.46, 789.12] |
| Annual ROI % (list) | Round each to **2 decimals** | [16.86543, 18.2319] → [16.87, 18.23] |
| Annual Rents $ (list) | Round each to **0 decimals** | [2318.55, 2388.1] → [2319, 2388] |

✅ 7. Chart Fixes in real\_estate\_new\_func\_git.py — Aligned y-axis keys to updated metric names for chart rendering.

✅ 8. Dual Property Comparison Feature

• Add UI functionality in Streamlit to allow uploading and comparing two sets of property inputs.  
• Layout should show side-by-side metrics for both properties (Cash Flow, ROI, CoC, etc.).  
• PDF export should include two tables or columns for comparison.  
• Changes will be required in:  
 - real\_estate\_new\_func\_git.py: add new input sections, comparison logic, chart updates.  
 - calculations.py: return metric dicts for both properties.  
 - pdf\_generator.py: render dual-column PDF output.

### Regarding 8: Breakdown: What the Dual Property Comparison Feature Encompasses

#### 💻 UI Layout (Streamlit)

* Display **two sets of input fields** side-by-side: Property A and Property B.
* On submit, show a **side-by-side results panel** for each:
  + Annual Cash Flow
  + Final Year ROI (%)
  + Cap Rate (%)
  + Monthly Mortgage
  + Grade / Verdict
* Add a **toggle**: Single vs Dual Property mode.

### Codebase Changes (File-by-File)

#### 1.

#### real\_estate\_new\_func\_git.py

* Add second input form for Property B.
* Add logic to:
  + Run calculate\_metrics(...) for both A and B.
  + Display both results in side-by-side st.columns(2).
* Update charting logic (optional) to compare both.

#### 2.

#### calculations.py

* Reuse calculate\_metrics(...) as-is — just call it twice:
  + metrics\_a = calculate\_metrics(...inputs...)
  + metrics\_b = calculate\_metrics(...other inputs...)

#### 3.

#### pdf\_generator.py

* Add dual-column table rendering:
  + Table([["Metric", "Property A", "Property B"]])
  + Include side-by-side comparison of key metrics.

## Render two AI Verdicts (A vs B). 🎯 Importance of Dual Comparison Feature for Your Tool

**Pros**

* Differentiator: It’s a feature users often want when deciding between multiple deals.
* Decision Aid: Users love seeing apples-to-apples metrics for two prospective investments.
* Upsell Potential: You can reserve it for paid tier, giving “free users” single deal only.

**Cons / Complexity**

* More UI complexity (two forms, layout, state management)
* More PDF complexity (two columns or tables)
* Must ensure metric naming consistency, error handling, missing data logic

Given that many competitors put this in paid versions, it aligns well with your tiered model: **include basic single‑deal in free version, reserve dual comparison for paid tier**.

So yes, it’s important as a premium feature — but it doesn’t necessarily need to be in MVP. You can plan it as your first “pro” upgrade.

## 🛠️ What Implementation Would Encompass (UI + Code Changes)

Here’s a more detailed breakdown:

### UI (in Streamlit / main / real\_estate\_new\_func\_git.py)

* Add a **checkbox / toggle**: “Compare Multiple Properties”
* If toggled, show **two sets of input forms** (Property A, Property B)
* Possibly use st.columns(2) for side-by-side layout.
* After inputs, compute metrics for both:
* Present results side-by-side:
  + Two metrics tables (or single table with 3 columns: Metric | A | B)
  + Chart overlay or dual line charts (or separate)
* On “Generate PDF”, pass both metrics and property\_data sets to pdf-generator.

### calculations.py

* No change to logic ideally — reuse same calculate\_metrics() function. You just call it twice.
* Make sure metric dict keys are consistent and stable (so pdf\_generator & UI reference the same keys).

### pdf\_generator.py

* Extend generate\_pdf(property\_data, metrics, summary\_text) (or define new wrapper) to accept **two metric dicts and two summary\_texts**.
* Render a **comparison table**:
  + Top row: headers (Metric, Property A, Property B)
  + Then rows: Cap Rate, Cash‑on‑Cash, ROI, Multi‑Year Cash Flow (side-by-side), etc.
* Maybe also render side-by-side “AI Verdict” + grade for both.
* Ensure skip\_keys, rename\_keys logic can handle two sets.

Architectural Path:

These values are now flowing end-to-end from calculations.py ➝ pdf\_generator.py ➝ Streamlit UI.

Gmail App pwd as of 09/29:

**fqjydyeujbbgiblm**

**ccpncedubzmfzssq**

### Why Use a Pseudonym?

| **Benefit** | **Explanation** |
| --- | --- |
| **Privacy** | You keep your real name and network out of public view. |
| **Freedom to experiment** | You can test rough ideas, unfinished features, or scrappy MVPs without risking your personal brand. |
| **No pressure** | You don’t feel exposed if no one replies — it’s part of testing, not failure. |
| **Honest feedback** | Strangers often respond more honestly when there’s no “personal” identity behind it. |

### How to Use a Pseudonym for MVP Feedback

| **Platform** | **How to Set It Up** |
| --- | --- |
| **Reddit** | Create a new Reddit account like @PropAnalyzerDev or @SmartInvestBot. No real name or email needed (use a burner if desired). |
| **Indie Hackers** | Create an account with a non-personal name like RentalROIBuilder. You don’t have to link a company or personal profile. |
| **Discord** | Join a real estate or maker community as @RE\_DealEval. |
| **Product Hunt (optional)** | Makers sometimes launch under a “team” name with no individual shown. |

### 🧵 Example Reddit Post Using Pseudonym

**Posted by**: u/SmartEvalTest

**Title**: 🏡 Looking for feedback — lightweight real estate deal evaluator (mobile-friendly, PDF export)

**Body**:

I’m testing a simple tool for modeling real estate ROI (buy-and-hold focus). It’s mobile-optimized, no login needed, and gives quick 10-year projections.

Here’s a private link + password:

🔗 https://real-estate-deal-evaluator.streamlit.app

🔑 Password: SmartInvest1!

Would love feedback from fellow investors or analysts — especially on:

* Clarity of cash-on-cash, final year ROI
* Missing features you’d want
* Overall usefulness

Will keep this live for a few days!