## **Developer Guide: Upgrading the NOI Analyzer Tool**

### **1. Overview**

We’re enhancing the existing NOI analyzer with a **complete upgrade package** that delivers PDF export functionality and a modern, visually appealing results dashboard and fix couple issues. You will:

1. Implement PDF report export in the analyzer backend.
2. **KPI Integration** — Compute and deliver core performance metrics alongside NOI analyses.
3. Polish the results page UI to match the attached design.
4. Ensure all styling, fonts, and components align with the provided mock.
5. Replace website’s logo with attached PNG file
6. Create dark / light theme option

### **2. KPI Engine Update**

**What is KPI Integration?** KPI integration embeds calculation of critical real estate metrics (EGI, OpEx Ratio, NOI Margin, GRM) into each NOI analysis request. Instead of manual spreadsheets, users get contextual insights automatically.

**Why integrate KPIs now?**

* **Actionable Context:** Users immediately understand efficiency and returns, not just raw NOI numbers.
* **Engagement:** Richer dashboards increase time-on-tool and perceived value.
* **Differentiation:** Sets us apart from basic variance calculators.

#### **2.1 Data & Formulas**

| **KPI** | **Formula** | **Data Required** |
| --- | --- | --- |
| **Effective Gross Income** | EGI = GPR - (Vacancy + Concessions + Bad Debt) | GPR, Vacancy, Concessions, Bad Debt |
| **Operating Expense Ratio** | OpEx Ratio = Operating Expenses / EGI | Operating Expenses, *EGI* |
| **NOI Margin** | NOI Margin = NOI / EGI | NOI, *EGI* |
| **Gross Rent Multiplier** | GRM = Property Value / GPR | Property Value, GPR |

#### **2.2 Ensure KPI Module Integration**

1. **Verify KPIEngine Availability:** Confirm app/services/kpi\_engine.py exists and exports generate\_kpis(data: dict) -> dict with keys: egi, operating\_expense\_ratio, noi\_margin, gross\_rent\_multiplier.

**API Payload Enhancement:** In your GET /properties/{id}/analysis endpoint, ensure the JSON response contains a top-level kpis object:  
  
 {

"financials": { /\* existing data \*/ },

"kpis": {

"egi": 12345.67,

"operating\_expense\_ratio": 0.45,

"noi\_margin": 0.40,

"gross\_rent\_multiplier": 11.5,

}

}

1. **Error Handling:** If any KPI value is null or missing, default to null in JSON so the frontend can display “--” placeholders.

#### **2.2 UI Hook for KPI Cards**

1. **Data Fetch:** In AnalysisPage.tsx, after loading analysis data, extract analysis.kpis.
2. **Component Integration:** Use the existing KpiCard component to render four cards:  
   * EGI
   * Operating Expense Ratio
   * NOI Margin
   * Gross Rent Multiplier
3. **Styling Fixture:** Ensure each card aligns with the dark-mode style: semi-transparent backgrounds, Inter/Roboto font, white text, green/red deltas.
4. **Positioning:** Place the KPI section immediately above or below the dashboard charts, as per the attached mock.

### **3. New Feature: PDF Report Export**

...

**Goal:** Let users download any analysis view as a professionally formatted PDF.

#### **2.1 Backend Changes**

1. **Install PDF Library:** Add WeasyPrint or ReportLab to your requirements.txt.
2. **Render HTML Template:** Create a Jinja2 template (templates/report.html) that mirrors the on-screen dashboard.

**PDF Endpoint:** In app/api/reports.py, add:  
  
 @router.get("/properties/{id}/report/pdf")

async def export\_pdf(property\_id: int):

data = get\_analysis(property\_id)

html = render\_template('report.html', \*\*data)

pdf = HTML(string=html).write\_pdf()

return Response(content=pdf, media\_type='application/pdf',

headers={

'Content-Disposition': f'attachment; filename="Property\_{property\_id}\_Report.pdf"'

})

1. **Link in Analysis Response:** Add pdf\_url to the JSON payload so frontend can show a “Download PDF” button.

#### **2.2 Frontend Integration**

**Download Button:** In the AnalysisPage.tsx, add a button next to the date picker:  
  
 <Button variant="outline" onClick={() => window.open(kpis.pdf\_url)}>

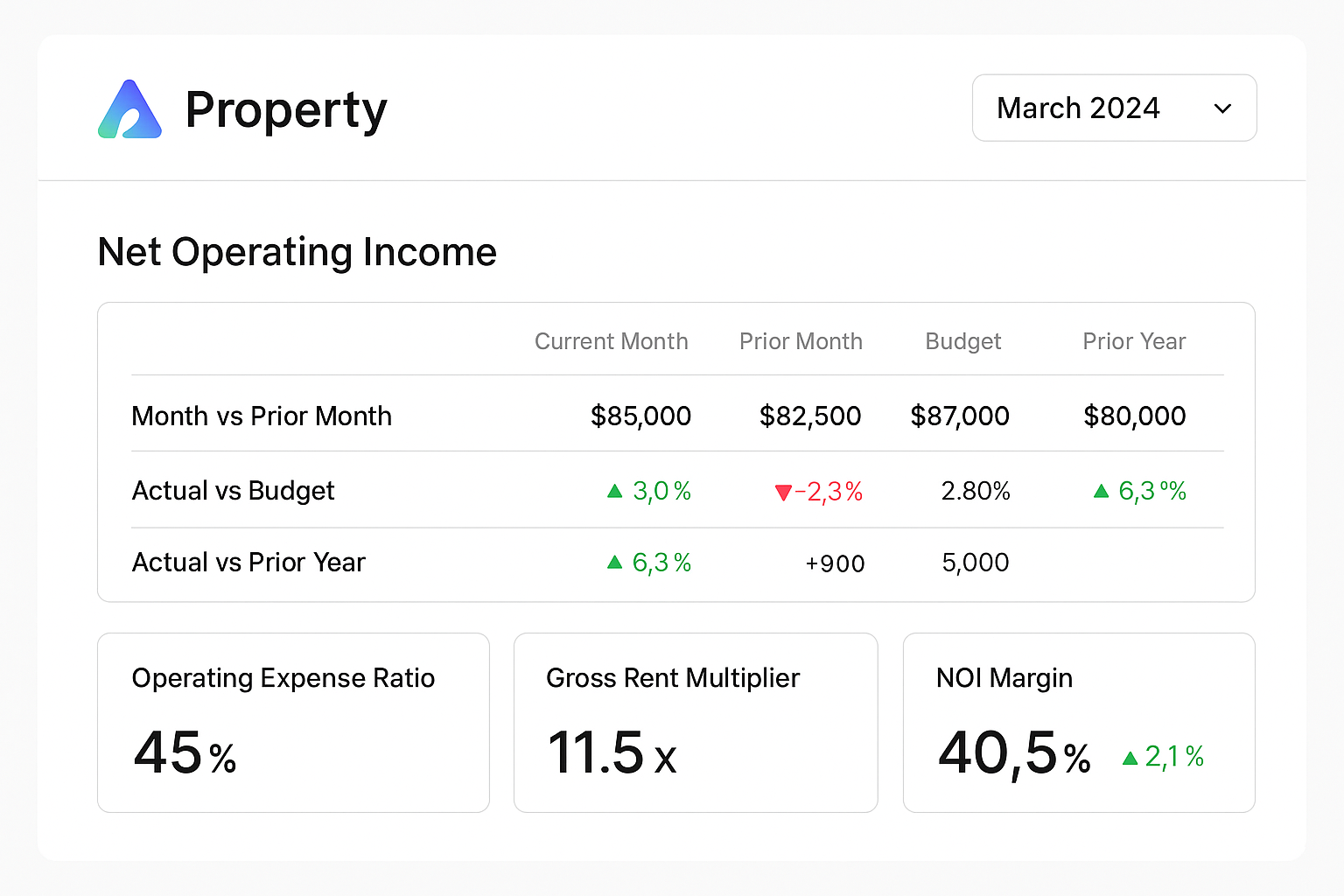
Download PDF

</Button>

1. **UX Tip:** Disable the button until the analysis has fully loaded.

### **3. Results Dashboard Redesign**

Use the attached images as the exact design spec for the results page. Below is a detailed breakdown:



#### **3.1 Layout & Style**

* **Background:** Very dark navy (#0A0F1E)
* **Cards:** Rounded 2xl corners, soft shadows, semi-transparent dark panels (#12203E with 50% opacity)
* **Font:** Use Inter or Roboto—white text at 16px for labels, 24px for values, 14px for axis labels
* **Primary Accent:** Electric blue (#3BA9FF) for lines, doughnut charts, bar charts
* **Secondary Accent:** Sky blue (#74C0FC) for secondary series

#### **3.2 Component Details**

1. **Net Operating Income Chart** (Top-left)  
   * Dual-line chart, legend below:  
     + Dark blue line for Month vs. Prior Month
     + Light blue line for Actual vs. Budget
   * X-axis: abbreviated months (Jun, Feb, Mar, …)
   * Y-axis: currency ticks on left (14k, 25k, 44k, 64k)
2. **Actual vs. Budget Doughnut** (Top-right)  
   * Single arc showing % complete (82% fill)
   * Large centered percentage text (48px)
3. **Actual vs. Prior Year Bar Chart** (Bottom-left)  
   * Vertical bars for each month
   * Negative values in darker shade
   * Axis labels in light gray (#B0B6C2)
4. **Variance Table** (Bottom-right)  
   * Simple table: three rows (Month vs Prior, Actual vs Budget, Actual vs Prior Year)
   * Columns: Description | % Delta | $ Value
   * Use 16px white for text, 14px gray (#95A1B3) for subtext

#### **3.3 KPI Cards Section**

Ensure styles match:

* **Card background:** #12203E @ 60% opacity
* **Metric label:** 14px, #E0E6ED
* **Value:** 32px, white
* **Delta arrow:** green (#4ADE80) for positive, red (#F87171) for negative

4. Replace website’s logo with the logo included in this PNG file (attached here, also will attach on conversation)



5. Create a tool that allows users to choose between dark theme and light theme.

### **4. Testing & Quality**

1. **Visual Regression:** Add storybook snapshots or Percy tests to lock down the new UI.
2. **PDF Accuracy:** Compare rendered PDF against the HTML dashboard visually.
3. **Cross-Browser:** Ensure charts and PDF generation work on Chrome, Firefox, Safari.