Value in Use (VIU) Automation Project Report

# Purpose

The Value in Use (VIU) calculation is a fundamental part of impairment testing under IAS 36, where the recoverable amount of an asset or cash-generating unit (CGU) is determined by estimating the present value of future cash flows. The traditional Excel-based workflow was becoming increasingly inefficient due to growing data complexity and regulatory scrutiny. This project aimed to automate the entire VIU calculation process using a Python-based solution that incorporates financial modeling, user-driven adjustments, and robust data validation—all wrapped in an intuitive UI to enable faster decision-making and regulatory compliance.

# Problem Statement

The traditional VIU process faced several challenges:

- \*\*Manual Workflow\*\*: Excel files were cluttered with interdependent formulas that were hard to audit or debug.

- \*\*High Risk of Errors\*\*: Small changes could propagate inconsistently due to formula breakage or outdated links.

- \*\*Slow Turnaround\*\*: Each review cycle involved hours of manual entry, review, and reconciliation.

- \*\*Lack of Scenario Analysis\*\*: Comparing different assumption sets or stress scenarios required duplicating files manually.

- \*\*Limited Audit Trail\*\*: Historical inputs and outputs were not logged systematically, creating issues during reviews.

# Solution Approach

The proposed solution was a modular, Python-based tool that automated VIU computation while adding transparency and scalability. Key components of the solution include:

1. \*\*Automated Discounted Cash Flow (DCF) Engine\*\*: Converts projected financial inputs into VIU through structured logic that supports overrides for assumptions like Tangible Equity, Tax Rates, and Discount Rates.

2. \*\*Scenario Builder and Sensitivity Framework\*\*: Allows users to apply assumption changes interactively, visualize results, and generate unique scenario IDs for audit purposes.

3. \*\*Dash-based User Interface\*\*: Enables user-friendly file uploads, table visualizations, scenario selection, and downloadable outputs.

4. \*\*Validation Layer\*\*: Automated range checks, YoY movement validations, and consistency checks ensure the integrity of input data.

5. \*\*Database-Backed Logging\*\*: All scenarios are logged with metadata such as timestamp, country, GB, and unique identifiers to maintain full traceability.

# Features

- \*\*Automated DCF Model\*\*: Calculates VIU using projected cash flows, TE, and discount factors.

- \*\*Scenario Builder\*\*: Create, edit, and compare user-defined scenarios on key assumptions.

- \*\*Sensitivity Analysis\*\*: Quantifies impact of changes in inputs like TE, tax, and PBT.

- \*\*Dash UI\*\*: Visualizes projections, inputs, and results interactively.

- \*\*Input Validation\*\*: Checks for anomalies, missing data, and large YoY changes.

- \*\*Database Logging\*\*: Logs all user configurations and outputs with metadata.

- \*\*Total Override Functionality\*\*: Aggregates across businesses to produce consolidated results.

# My Contribution in Features

- \*\*DCF Engine\*\*: Developed and tested the FCFE projection logic with multiple override conditions.

- \*\*Scenario Builder\*\*: Designed the structure for sensitivity inputs and integrated it with Dash UI.

- \*\*Adjustment Tab\*\*: Implemented the logic for applying adjustments on top of original config values.

- \*\*Total Override Functionality\*\*: Created function to update 'Total' business line dynamically using the latest config of other businesses.

- \*\*Database Logging\*\*: Developed mechanism to log each config with unique ID, timestamp, country, and GB.

- \*\*Dash UI\*\*: Built callbacks to render scenario accordions and manage data flow between tabs.

- \*\*Data Quality Checks\*\*: Integrated rule-based validations and automated discrepancy detection.

# Current Status

The project has reached a stable working version with the following status:

- ✅ \*\*Completed\*\*: Core engine for VIU computation, interactive UI for input/output handling, scenario builder and sensitivity module, logging and audit trail.

- 🔄 \*\*In Progress\*\*: UI/UX refinement for better responsiveness, dynamic export to Excel with formatted outputs and scenario summaries.

- 🚀 \*\*Planned Enhancements\*\*:

- Integration with enterprise databases for automatic data refresh cycles.

- Role-based access control for user management.

- Email automation for reporting impairments or major deviations.

# Learnings

- 📘 \*\*Accounting Expertise\*\*: Developed a deeper understanding of IAS 36 requirements and how VIU fits into the impairment review cycle.

- 💻 \*\*Software Engineering\*\*: Learned how to structure a full-stack Python application with modular back-end and dynamic front-end capabilities.

- 🧩 \*\*Data Engineering\*\*: Worked with complex Excel templates and translated them into consistent and validated data pipelines.

- 🛠️ \*\*Performance Tuning\*\*: Optimized data handling and ensured low memory footprint while processing high-volume Excel files.

- 🤝 \*\*Stakeholder Engagement\*\*: Collaborated with finance controllers, risk managers, and model validators to ensure expectations were met.

- 🧾 \*\*Audit-Readiness\*\*: Created a logging framework that supports easy reproducibility and compliance audits.