FP&A Backend Documentation

This document outlines the main logical modules of the FP&A Backend Super v3. The application is built using the FastAPI framework for high-performance API endpoints and Pydantic for robust data validation.

The modules are grouped by their functional purpose within the FP&A (Financial Planning & Analysis) workflow.

Core Technologies

FastAPI: The web framework used to create, route, and serve all API endpoints. It's initialized as the api object.

Pydantic: Used to define data "models" (e.g., BvARequest, HeadcountRow). These models automatically validate incoming request data to ensure it matches the expected format and data types.

Application Modules

1. BvA (Budget vs. Actual) Module

Description: This module handles Budget vs. Actual (BvA) variance analysis. Its primary function is to compare financial results against the plan and generate insights.

Endpoints:

POST /bva/analyze: Receives a year and scenario (e.g., "Actuals" vs "Budget"). It returns a summary analysis, including an "AI Narrative" and a list of "hotspots" or key variances.

2. Headcount Module

Description: Manages all employee and salary-related data. This includes maintaining the employee roster and calculating the financial impact of headcount (salaries, taxes, benefits) on the budget.

Endpoints:

GET /headcount/list: Retrieves the complete list of all employees currently in the system.

POST /headcount/upsert: Adds a new employee or updates an existing employee's details (like salary or start date) using the HeadcountRow model.

POST /headcount/bake_to_budget: A process-heavy endpoint that takes the current headcount roster and calculates all associated financial expenses, "baking" them into the main budget model.

3. Solver Module

Description: Provides advanced financial modeling tools. This module is used for complex calculations that "solve" for a specific target.

Endpoints:

POST /solver/goal_seek: Performs a goal-seek analysis. For example, it could find the required "percent change" in revenue needed to hit a specific profit margin target.

4. Scenario Module
Description: Manages the creation and manipulation of different financial scenarios (e.g., "Baseline," "Upside Case," "Downside Case").

Endpoints:

POST /scenario/clone: Creates a new scenario by copying an existing one.

POST /scenario/sensitivity: Runs a sensitivity analysis, likely adjusting key drivers (e.g., "increase COGS by 5%") and calculating the impact across the model.

5. Versions Module
Description: Handles version control for scenarios. This allows users to save "checkpoints" of their work and revert to them later, preventing data loss during modeling.

Endpoints:

POST /versions/save: Saves the current state of a scenario as a new, named version.

GET /versions/list: Lists all available saved versions for a given scenario.

POST /versions/restore: Reverts a scenario's data back to a selected saved version.

6. Excel (Data I/O) Module
Description: Manages the bulk retrieval and submission of financial data, typically to and from a web-based spreadsheet or data grid interface (often referred to as "Excel-like").

Endpoints:

POST /excel/retrieve: Fetches a large set of raw financial data (e.g., P&L lines by month) for display in a grid.

POST /excel/submit: Receives a payload of data (e.g., manual inputs or adjustments from a grid) and writes those changes to the database.

7. Boardpack (Reporting) Module

Description: Responsible for generating high-level, presentation-ready reports, such as those for a board meeting.

Endpoints:

GET /boardpack/generate: Kicks off a process to generate a board pack (likely a .pptx file) based on a specific year and scenario.