# Project Progress Report

**Project Name: Smart Financial Advisory (SFA_V5)**

**Current Date: 2025-12-07**

## Backend Overview

The backend is built on FastAPI, structured around a modular Multi-Agent System (RAMAS). It utilizes a Planner-Worker-Auditor agent flow. Recent changes include a successful refactor to remove MongoDB dependencies, fully transitioning to SQLite for financial data storage. Implemented specific tools for SQL generation (sql_tools.py) and robust error handling in the analytics endpoints. Infrastructure uses uvicorn for serving the app and groq for LLM inference.

## Dataset Description

The project primarily deals with US SEC financial data. Structured Data is stored in financial_data.db, containing detailed financial reports (10-K/10-Q) linked to company submissions. Unstructured textual financial documents are processed and indexed into a ChromaDB vector store for RAG operations.

## Database Structure

The system relies on a relational SQLite schema (financial_data.db):
- Table 'submissions': Stores company metadata (CIK, Name, Clean Name, SIC code, Country).
- Table 'numbers': Stores the raw financial facts (Tags, Values, Dates, Units of Measure).
- Table 'users' (in users_accounts_data.db): Manages authentication, roles (Admin/Manager), and password hashes.

## Frontend Framework

Tech Stack includes Native HTML5, CSS3, and JavaScript integration using Jinja2 templates served by FastAPI. It uses Bootstrap 5 for responsive layout and styling. Plotly.js is implemented for rendering dynamic, interactive financial charts (Revenue & Net Income trends) on the manager dashboard.

## Agents Used

The system implements a Retrieval-Augmented Multi-Agent System (RAMAS):
1. Planner Agent: Decomposes complex financial queries into logical steps.
2. Worker Agent: Executes tools (SQL generation or Vector Search) to retrieve data.
3. Auditor Agent: Verifies the worker's output and formats the final response for the user.

Special Logic: A 'Chain-of-Tables' reasoning engine is implemented in backend/llm.py.

## Testing Methods

Framework: pytest is used for unit and integration testing.
Scripts exist for different layers:
- tests/test_all_sql.py: Verifies SQL query generation and database connectivity.
- tests/test_live.py: Simulates live environment interactions.
- verify_chatbot.py: Tests the end-to-end chat flow.

# Project Progress Report

## Verification Techniques

Manual Verification: A 'Golden Dataset' approach is outlined to manually grade model responses against known answers.
Automated Checks: The sql_tools.py module includes safety checks to prevent destructive SQL commands (only SELECT allowed).
Visual Validation: The dashboard graphs serve as a visual verification step for the data aggregation logic.

## Models Implemented

LLM: Llama-3.3-70b-versatile (via Groq API).
Purpose: Handles natural language understanding, SQL generation, and final answer synthesis.
Embedding Model: Used within the RAG pipeline for vectorizing financial documents (specific model abstracted in rag.py).

## API Details

The application exposes a RESTful API via FastAPI:
- POST /api/auth/token: JWT-based authentication.
- GET /api/dashboard/metrics: Delivers aggregated financial metrics (Assets, Revenue) and trend data for graphs.
- POST /chat: The main interface for the AI financial advisor.

## Accuracy Metrics

Current Status: Accuracy is currently measured by the successful execution rate of generated SQL queries and the non-emptiness of retrieval results.
Next Steps: Formalize an automated evaluation pipeline using the 'Golden Dataset' to generate a quantitative accuracy score (e.g., % of queries correctly answered).

## Summary

The SFA_V5 project has successfully established a robust, local-first architecture using SQLite and FastAPI. The core 'Chain-of-Tables' reasoning and RAG pipelines are functional, and the frontend is integrated with dynamic visualization tools. The immediate focus is on refining the accuracy of the SQL generation agent and expanding test coverage to ensure reliability across a wider range of financial queries.