Macroeconomic Insights & Portfolio Response Dashboard: A Research-Driven Visualization Project

Project Objective

This project develops an interactive, research-informed dashboard that models the relationship between key macroeconomic indicators and the performance of financial asset classes. It integrates academic and industry research, quantitative modeling, and visual analytics to provide actionable insights for both retail and institutional investors.

The dashboard includes real-time and historical macroeconomic data (e.g., GDP, inflation, interest rates, ΔCLI) and simulates asset class responses (e.g., equities, bonds, commodities, FX) under various macroeconomic scenarios. It also features predictive modules, uncertainty overlays, and an educational component grounded in macroeconomic research.

Project Status

This project is conducted independently by: **Triumph Kia Teh**
Computer Science Modified with Economics major and Statistics minor (Quantitative Research focus)

Anticipated Impact

- Demonstrates integration of academic theory, quantitative modeling, and financial intuition.
- Provides a scalable tool for macro-asset forecasting and education.
- Serves as a publication-ready, portfolio-worthy deliverable.

Evaluation Criteria

- Technical accuracy and robustness of models
- Quality of research integration and sourcing
- Usability and clarity of the dashboard interface
- Originality and insightfulness of asset response mapping

- Clarity and rigor in final documentation/reporting

Motivation and Relevance

Discipline	Motivation
Quantitative Finance	Build data-driven macro forecasting tools using real indicators (ΔCLI, GDP surprises, inflation).
Macroeconomics	Bridge theory and empirical models to forecast asset sensitivity to business cycles.
Markets & Investing	Highlight the role of macro trends in global asset allocation strategies.

SMART Framework

- **Specific**: Build an interactive macroeconomic dashboard informed by research, supporting real-time scenario modeling and visual analysis.
- **Measurable**: Deliver 6 milestone phases with key outputs: 1 research digest, 1 structured DB, 4 dashboards, 1 predictive module, 1 GitHub repo.
- **Achievable**: Utilize free-tier APIs/tools (FRED, Yahoo Finance, Streamlit Cloud) and independent programming skills.
- **Relevant**: Aligns with career goals in forecasting, data science, and quantitative research.
- **Time-Bound**: Completed in 12 weeks, with clear weekly milestones.

Chosen Research Backbone

- **Di Bonaventura & Morini (2024)** – *Macro-Financial Factors and Asset Classes*
  Guides scenario engine logic and macro filter design.
- **Macrosynergy (2024)** – *Macroeconomic Trends and Financial Markets*
  Informs multi-asset module expansion and macro signal processing.
- **Long et al. (2022)** – *ΔCLI and Global Stock Returns*
  Validates predictive modeling via leading indicators (CLI).

Phase-by-Phase Timeline

📀 Phase 1: Setup, Research Review, Data Collection (Weeks 1–2)

- Set up PostgreSQL schema and API pipelines (FRED, World Bank, Yahoo Finance)
- Normalize initial asset class time series
- Compile literature and build initial Research Digest

Deliverables:

- Raw data repository
- Research Digest (3–5 pages)
- Exploratory visuals (e.g., Tableau or Altair)

📀 Phase 2: Data Cleaning, Structuring, Research Alignment (Week 3)

- Structure datasets by date frequency
- Clean, fill, and normalize time series
- Build core ETL pipeline using Pandas and SQLAlchemy

Deliverables:

- Structured PostgreSQL database
- Annotated schema with research tags
- Updated research digest

📀 Phase 3: Dashboard Development & Scenario Mapping (Weeks 4–5)

- Develop interactive dashboards (Streamlit)
- Integrate dropdown filters and user inputs
- Connect visual panels to supporting research logic

Deliverables:

- Streamlit dashboard (2+ panels)
- KPI tracker and interactive selector
- Academic blurbs for each indicator

📀 Phase 4: Correlation Analysis & Research Validation (Weeks 6–7)

- Compute Pearson and Spearman matrices
- Build correlation and time-lag visualizations
- Write comparative analysis matching to literature

Deliverables:

- Interactive correlation dashboard
- Validation report (2 pages) with citations

📀 **Phase 5: Deployment & Interface Design (Weeks 8–9)**

- Deploy on Streamlit Cloud and GitHub Pages
- Automate refresh (cronjob or Streamlit-triggered)
- Add user documentation and inline guidance

Deliverables:

- Live deployed app
- Instructional sidebars and blurbs
- GitHub repo with code + documentation

📀 **Phase 6: Forecasting Models & Predictive Interface (Weeks 10–12)**

- Implement ARIMA and linear regression modules
- Build scenario simulation interface (e.g., ΔCLI $\rightarrow$ asset performance)
- Frame outputs using real macroeconomic events

Deliverables:

- Predictive analytics module
- Scenario walkthroughs
- Modeling summary with citations

Final Deliverables

- PDF: Research Digest

- Live Web App: Streamlit Dashboard
- GitHub Repository:
    - Cleaned source code and ETL scripts
    - Setup instructions and usage guide
    - APA-style academic references
    - Deployment links to dashboards and tools

This plan reflects a research project that merges data science, macroeconomic analysis, and applied quantitative research. It ensures academic rigor and professional presentation, suitable for both faculty evaluation and industry portfolio use.

Detailed Action Report

1. Project Overview & Objectives

1. **Goal**

    - Develop a research-driven visualization tool integrating macroeconomic data and financial assets, showcasing how indicators (e.g. GDP, CPI, CLI, etc.) influence various asset classes over time.

2. **Objectives**

    - Implement an interactive Streamlit dashboard that retrieves data from publicly available APIs (FRED, Yahoo Finance, World Bank) without relying on local or hosted databases.

- Provide scenario-based insights for major asset classes (equities, bonds, commodities) informed by the latest research.

- Enable “refresh” capabilities so end-users can retrieve updated data on demand in an ephemeral environment (i.e., Streamlit Cloud).

3. **End-User Benefit**

   - **Researchers / Students** can quickly see real-world macro correlations.

   - **Investors / Professionals** can reference state-dependent macro factor effects on portfolio allocations.

   - **Public** can explore leading indicators, job data, inflation, and how they align with asset performance.

---

2. Summary of Implementation Steps

2.1. Initial DB-Driven Prototype (Local / Postgres)

- **Original Approach**:

  1. Set up PostgreSQL locally for raw data ingestion (FRED, Yahoo Finance, World Bank).

  2. Created ETL scripts in `scripts/db/` (e.g. `insert_fred_data.py`, `etl.py`) to load tables.

  3. `app.py` used `pd.read_sql(...)` to query from Postgres.

- **Limitations**:

    1. Required local database running 24/7 or hosted DB (cost, complexity).

    2. Tied to personal machine or paid DB solution.

2.2. Transition to an “API + CSV” Approach

- **Rationale**:

    1. Deploying to Streamlit Cloud with ephemeral storage means no persistent DB needed.

    2. CSV-based approach is simpler: fetch from APIs, store ephemeral files in `data/raw/`.

    3. Freed from local DB references → fully portable, easy to share publicly.

- **Main Changes**:

    1. Removed or disabled `db_connect.py` & references to `engine` in the final `app.py`.

    2. Adopted direct API calls in `scripts/data_pipeline/fetch_*.py`, writing to CSV under `data/raw/`.

    3. Updated `app.py` to load from CSV files via new helper functions `load_fred_csv()`, `load_yahoo_csv()`, `load_worldbank_csv()`.

2.3. Final Streamlit Application

- **Core Files**:

    1. `dashboards/streamlit_app/app.py` → Main Streamlit code.

    2. `scripts/data_pipeline/fetch_fred_data.py`, `fetch_yahoo_data.py`, `fetch_worldbank_data.py` → Scripts that fetch from FRED, Yahoo Finance, World Bank, saving CSV.

    3. `data/processed/pearson_correlation_matrix.csv`, `spearman_correlation_matrix.csv` → For correlation matrix display.

    4. `app.py` includes an optional “Refresh All Data” button to re-run those scripts and update local CSV.

- **Key Features**:

    1. **KPI Section:** Latest CPI, GDP, Unemployment, CLI.

    2. **FRED Macro Indicator Trends:** Filter, smoothing, forecasting.

    3. **Yahoo Finance Asset Correlation:** Merge macro data with assets (sp500, gold, bond10y, oil, eurusd, reit_etf).

    4. **World Bank Indicators:** Display long-term stats, e.g. population, inflation, etc.

    5. **Correlation Matrix Explorer:** Show Pearson or Spearman correlation among selected fields.

    6. **Research References & Summaries** from Long et al. (2022), Macrosynergy (2024), Di Bonaventura & Morini (2024).

- **Ephemeral Refresh**:

1. Each “Refresh” call triggers `subprocess.run(...)` on your `fetch_*.py` scripts, overwriting CSV files.

2. `st.cache_data.clear()` ensures reloaded data.

3. All changes are ephemeral; once the container restarts, the app defaults back to baseline or re-fetches.

---

3. Technical Details

1. **Folder Structure**

    - **`data/raw/fred/`**: Contains `fred_cpiaucns.csv`, `fred_gdp.csv`, `fred_unrate.csv`, `fred_usslind.csv`.

    - **`data/raw/yahoo/`**: Contains `sp500.csv`, `gold.csv`, etc.

    - **`data/raw/worldbank/`**: Contains `worldbank_us_macro.csv`.

    - **`data/processed/`**: Contains correlation matrix CSVs.

2. **Fetch Scripts** (under `scripts/data_pipeline/`):

    - `fetch_fred_data.py`: Calls `utils_fred.py` to get FRED series → saves CSV in `data/raw/fred/`.

    - `fetch_worldbank_data.py`: Uses `pandas_datareader.wb` to get US macro → saves `worldbank_us_macro.csv`.

- `fetch_yahoo_data.py`: Uses `yfinance` to fetch sp500, gold, etc. → saves in `data/raw/yahoo/`.

3. `app.py`
    - **Imports**: altair, yfinance, etc.
    - **Sections**: KPI, FRED macro trend, Yahoo correlation, World Bank metrics, Correlation matrix.
    - **Refresh**: A button in the sidebar calls the fetch scripts.
    - **Auto-Refresh**: (Optional) Checking if last fetch was >4 hrs ago & daily limit not exceeded.

---

4. Deployment Steps

1. **Push to GitHub**: Ensure all scripts, `requirements.txt`, and final `app.py` are in the repo.
2. **Go to share.streamlit.io** → "New app" → select GitHub repo & branch → pick `app.py` path.
3. **Deploy**: Streamlit Cloud builds environment.
4. **Use**: The final app is live at a unique URL, ephemeral CSV changes are possible.

---

5. Known Constraints & Next Steps

- **Ephemeral CSV**:

    1. Each time the Streamlit Cloud container restarts (after inactivity or new deploy), the CSV changes from the previous session vanish. Users can always click “Refresh All Data” to fetch anew.

- **Possible Enhancements**:

    1. **Persistent Storage**: Use S3 or Google Drive APIs if you want to preserve updated CSV across sessions.

    2. **Scheduled Refresh**: If you want truly “auto” refresh daily, you might rely on GitHub Actions or an external schedule.

    3. **Greater Interactivity**: Additional scenario modeling or user inputs for deeper analysis.

    4. **Security**: If you have sensitive API keys, store them in Streamlit Secrets or environment variables.

---

6. Conclusion

This final version of the “Macroeconomic Insights & Portfolio Response Dashboard” accomplishes the project’s core objective:

- **API-Driven** → No local or external DB hosting required.

- **Flexible** → Simple CSV-based ephemeral storage, refreshed on demand.

- **Informative** → Provides a broad multi-asset, multi-indicator perspective, consistent with academic research.

By decoupling from local Postgres usage, the application is fully deployable on Streamlit Cloud, letting you share it with colleagues, students, or the public for real-time or near-real-time macroeconomic exploration.