Project: Python, Git, Linux for Finance

Scenario

You are part of the quantitative research team in an asset management company in Paris. Your mission is to support the fundamental portfolio managers with quantitative tools. Management has asked you to design a professional-looking dashboard that can:

- Continuously retrieve and display financial data in real time,
- Implement quantitative strategies and portfolio simulations,
- Clearly show results and key metrics in an interactive and user-friendly way.

You will work as a two-person team, collaborating through GitHub and deploying your code on a Linux server. Each team member is responsible for a specific module (single asset analysis or portfolio analysis), but the final result must be a single integrated platform.

Objective

Build and deploy an online platform that retrieves financial (or economic) data in real time, displays it in an interactive dashboard, and provides tools for backtesting and portfolio analysis. The project must simulate a professional workflow: students will collaborate using Git (GitHub), run their code on a Linux virtual machine, and structure their app using Python (with Streamlit or an equivalent framework).

Core Features

1. Retrieve data from a dynamic source (e.g., market prices, crypto, macroeconomic indicators) that updates at least every few minutes.
2. Data can be retrieved either through:
   - a public API
   - web scraping using Bash, grep, regex, or Python libraries
3. Display the **current value** of the chosen asset(s) on the dashboard.
4. Plot a **main time series graph** of the data, combining raw values and strategy results (see Division of Work for details).
5. Automatically refresh data every **5 minutes**.
6. Provide a daily report (e.g., volatility, open/close price, max drawdown) generated at a fixed time (e.g., 8pm) via **cron**, stored locally on the VM. The **cron job configuration and scripts** must also be included and documented in the GitHub repository.

7. Ensure the app is always running (24/7) on the hosted Linux machine, while minimizing cloud costs.

8. The GitHub repository must be well structured: clean code, informative commit history, clear README.

Division of Work

This project must be completed by a group of exactly two students. Each student is responsible for a separate module of the platform. Both modules must include back-end (data handling, calculations) and front-end (dashboard visualization, user interaction) components.

- **Quant A — Univariate - Single Asset Analysis Module**
  - Focus on one main asset at a time (e.g., ENGI, EUR/USD, gold).
  - Implement at least two backtesting strategies (e.g., buy-and-hold, momentum, etc.).
  - Display performance metrics: max-drawdown, Sharpe ratio, etc.
  - Provide interactive controls (strategy parameters, periodicity selection).
  - The main chart must show the raw asset price together with the cumulative value of the chosen strategy (two curves should appear on the graph).
  - **Optional Bonus**: Implement a simple predictive model (e.g., linear regression, ARIMA, ML regression, or tree-based model) to forecast future values. Predictions could include confidence intervals and be visualized alongside historical data.
- **Quant B — Multivariate - Multi-Asset Portfolio Module**
  - Extend the app to multiple assets (at least 3 different assets at the same time).
  - Implement portfolio simulation (e.g., equal weight, custom weights).
  - Display portfolio metrics: correlation matrix, diversification effects, portfolio volatility and returns.
  - Provide user access to portfolio strategy parameters (e.g., rebalancing frequency, asset weights, allocation rules).
  - Provide visual comparisons between single assets and the portfolio.
  - The main chart must show multiple asset prices together with the cumulative value of the portfolio.

Collaboration Rules

- Both students must work on their module independently, but integration is required for a single functioning dashboard.
- Proper use of GitHub is mandatory:
  - Separate branches for each module.
  - Pull requests and conflict resolution are part of the workflow.
  - Separation of the two tasks should be respected (checked in commits history). Severely downgraded if not respected.
- Project details must also be submitted via this Google Form.
- No external help is allowed beyond online documentation, forums and AIs.

Evaluation Criteria

- Functionality: the online interactive platform must be accessible and fully operational during the evaluation week.
- Code quality: clean, documented, modular, easy to debug.
- GitHub usage: clean repository, detailed README, informative and clean commits, correct use of branches.
- Dashboard quality: user-friendliness, professional design, clear visualization of results.
- Robustness: app must handle errors gracefully (e.g., failed API call should not crash the app).
- Bonus points will be awarded if the project retrieves data using the **Bloomberg API**.
- Bonus points for optional ML prediction features.

Example Data Sources

- https://www.investing.com
- https://coinmarketcap.com
- https://tradingeconomics.com
- https://openweathermap.org
- https://finnhub.io
- https://fred.stlouisfed.org
- https://www.boursorama.com