Kalpi Capital — Quant Developer / Quant Researcher Assignment

Objective

Build a **Portfolio Optimization Engine** similar in functionality and user flow to:
https://www.portfoliovisualizer.com/optimize-portfolio#analysisResults

You must implement a complete end-to-end system including:

- Python code (functions + classes)
- A small FastAPI backend with endpoints
- A minimal UI (Streamlit or simple web page) to input parameters and view results
- Documentation explaining how to run it

---

1. Core Requirements

A. Portfolio Optimization Methods (Mandatory)

Implement the following portfolio optimizers:

1. **Mean–Variance Optimization (MVO)**
    - Generate efficient frontier
    - Maximum Sharpe portfolio
    - Minimum variance portfolio
2. **Conditional Value-at-Risk (CVaR)** minimization
3. **Risk Parity**
    - Equal risk contribution portfolio
4. **Tracking Error Minimization**
    - Against a selected benchmark
5. **Information Ratio Maximization**
    - Against a selected benchmark
6. **Kelly Criterion Optimized Portfolio**
    - Maximize expected geometric growth
7. **Sortino Ratio Maximization**
    - With user-defined Minimum Acceptable Return (MAR)
8. **Omega Ratio Maximization**
    - With user-defined MAR

9. **Minimum Maximum Drawdown Portfolio**
    - Optionally include MAR constraint

---

2. Inputs (UI + API)

User must be able to provide:

- Asset list (tickers)
- Date range (start, end)
- Method selection (one of the 9 optimizers)
- Risk-free rate
- MAR (for Sortino/Omega)
- Confidence level (for CVaR)
- Benchmark ticker (for tracking error / information ratio)
- Constraints:
    - Long-only or long-short
    - Min/max weight per asset
    - Sum of weights = 1 toggle
- Number of points on efficient frontier (for MVO)

---

3. Outputs (UI + API)

For each optimization:

- Optimized weights (table + JSON)
  Portfolio performance metrics:
    - Expected return
    - Volatility
    - Sharpe
    - Sortino
    - Omega
    - CVaR
    - Maximum drawdown
    - Tracking error (if relevant)
    - Information ratio (if relevant)
- Visualizations:
    - **Efficient frontier** (with chosen portfolio highlighted)
    - **Risk contribution plot** (for risk parity)
    - **Cumulative return chart** (in-sample)

---

4. Architecture Requirements

A. Python Code

- Use modular structure with classes like:
  `MeanVarianceOptimizer`,
  `CvarOptimizer`,
  `RiskParityOptimizer`,
  etc.

- Each class must have a clear `optimize()` method returning:

    - weights
    - metrics dict
    - diagnostics (optional)

B. FastAPI Backend

Expose endpoints:

1. `POST /optimize`
2. `GET /frontier` (for MVO)
3. `POST /load-data` (load historical data for selected tickers)

C. UI Layer

A simple UI (Streamlit or web page) that allows users to:

- Select method + parameters
- Run optimization
- See tables + charts like PortfolioVisualizer

---

5. Data

Use **any open data source** for Indian stocks.
Historical OHLCV + adjusted close is enough.

---

6. Deliverables

Submit:

1. **Final code** (Python + FastAPI + UI)
2. **Documentation** (README explaining how to run)
3. **Visual results** (screenshots or plots inside repo)
4. **Optional**: Deployed link or Dockerfile (adds points)

---

7. Brownie Points (Optional Extras)

- Add transaction cost or turnover constraints
- Add L1 sparse portfolio option
- Add multi-period backtest tab
- Add factor exposure report
- Add downloadable CSV/PDF of results

---

8. Contact & Deadline

For doubts:
📧 ashwar.gupta@kalpicapital.com
📞 +91-8871911901

Deadline: 1 day from the time you receive this assignment.
Early submission = extra points.