**ASG Microfund Project Plan: Building a Simulated Quant Fund**

**Overview**

This project aims to build a simulated (or very small capital) investment fund, structured and operated like a professional quant fund. The objective is to apply finance, computer science, and math skills to design, test, manage, and report on a multi-strategy portfolio.

This project will serve as a hybrid between a startup and a skill development platform, and can also become a standout personal portfolio/resume piece.

**Project Goals**

* Create a multi-strategy quant investment simulation.
* Backtest and paper trade strategies.
* Build a professional-grade dashboard and report system.
* Gain experience in alpha generation, risk management, and portfolio construction.
* Develop code, documentation, and analytics to showcase.

**Components**

**1. Strategy Development:** Develop 2-3 core quantitative strategies (unsure which to choose for now, will decide soon but these are potential placeholders):

* **Time Series Momentum**
  + Bias: Trend
  + Time Horizon: Medium
  + Market Condition Suitability: Trending/Breakout Market
* **Mean Reversion** (e.g., SMA reversion, Bollinger bands)
  + Bias: Contrarian
  + Time Horizon: Short
  + Market Condition Suitability: Sideways/volatile
* **Factor Investing** (e.g., quality, value, size, volatility)
  + Bias: Fundamental
  + Time Horizon: Long
  + Market Condition Suitability: Stable/Fundamentally-driven

**2. Data Management**

* Historical price data: yfinance, Alpha Vantage, or Quandl
* Factor data: Fama-French via Ken French Data Library
* Macroeconomic: FRED or yfinance

**3. Backtesting Engine (unsure which one to use)**

* bt for strategy testing and portfolio analysis
* backtrader for more granular, customizable backtests
* zipline (optional, if historical trade simulation is needed)
* possibly even backtesting.py

**4. Portfolio Construction**

* Equal weight
* Risk-parity
* Factor score weighted
* Volatility targeting

**5. Execution Simulation**

* Trade cost modeling (slippage, commissions)
* Rebalancing logic (monthly, weekly, trigger-based)
* Realistic trade simulation

**6. Risk Management**

* Max drawdown tracking
* Value at Risk (VaR)
* Stop-loss and position limits
* Rolling Sharpe ratio

**7. Performance Tracking**

* Portfolio value and NAV
* Benchmark comparison (e.g., S&P 500)
* Rolling returns, alpha, beta
* Strategy attribution

**8. Reporting System**

* Monthly PDF reports (returns, stats, commentary)
* Performance dashboard (Streamlit)
* Portfolio allocation charts
* Trade logs and metrics

**9. Research Documentation**

* Strategy notes and assumptions
* Backtest results and parameter tuning
* Risk/reward assessment
* Future improvements
* Use internal documents to build and make notes

**Technology Stack (not finalized)**

| **Module** | **Tools** |
| --- | --- |
| Data | yfinance, Alpha Vantage, FRED, Quandl |
| Backtesting | bt, backtrader, zipline |
| Analysis & Stats | pandas, numpy, scipy, statsmodels, quantstats, empyrical, pyfolio |
| Visualization | matplotlib, seaborn, plotly |
| Dashboard/Reporting | Streamlit, Jupyter, WeasyPrint for PDFs |

**Deliverables**

1. **Strategy Code** (modularized Python scripts)
2. **Backtest Reports** (Jupyter notebooks or exported PDFs)
3. **Dashboard** (Streamlit or local server app)
4. **Monthly Reports** (PDF format, auto-generated)
5. **README and GitHub Repo** (clean documentation)
6. **Research Notes** (strategy rationale and learning logs)
7. **Simulated Trading Journal** (CSV, SQLite, or Google Sheets)

**Milestone Roadmap (8-12 Weeks)**

| **Week** | **Focus Area** | **Key Outcomes** |
| --- | --- | --- |
| 1–2 | Infrastructure + First Strategy | End-to-end working backtest of a mean reversion strategy |
| 3–4 | Portfolio Logic + Strategy 2 | Multi-strategy support, benchmark, and risk stats |
| 5–6 | Reporting + Dashboard | PDF reports + Streamlit dashboard + trade journal |
| 7–8 | Final Strategy + Polish | Third strategy, cleaned repo, publish-ready |
| 9–10 | Publishing + Resume Packaging | Demo video, README polish, LinkedIn/GitHub update |

**Timeline**

**Week 1: Setup and Mean Reversion Strategy**

**Goal:** Set up project structure and run an end-to-end backtest of a mean reversion strategy.

**Files to complete:**

* requirements.txt
* main.py
* data/data\_loader.py
* strategies/mean\_reversion.py
* backtesting/engine.py

**Tasks:**

* Set up the GitHub repository and Python virtual environment.
* Create the full project folder structure.
* Implement a mean reversion strategy using simple SMA crossover logic.
* Build a basic backtesting engine that accepts strategy logic.
* Build data loader using yfinance.
* Run the backtest and output basic metrics to the console.

**GitHub Upload:**

* Initial commit with full project structure.
* Upload working mean reversion strategy and backtest results.
* Push test output to notebooks/.

**Week 2: Strategy Testing and Reporting Foundations**

**Goal:** Expand the first strategy with parameters and start building the reporting system.

**Files to complete:**

* portfolio/benchmark.py
* Expand strategies/mean\_reversion.py
* First version of reporting/generate\_report.py

**Tasks:**

* Add benchmark (e.g., S&P 500) to compare strategy performance.
* Output key metrics: Sharpe ratio, max drawdown, CAGR.
* Build visualizations: equity curve, drawdown, rolling Sharpe.
* Begin laying out a function for PDF report generation (using WeasyPrint or matplotlib exports).

**GitHub Upload:**

* Upload updated strategy results.
* Include benchmark plots in notebooks/.

**Week 3: Add Second Strategy and Portfolio Construction**

**Goal:** Add a second strategy and build a combined portfolio framework.

**Files to complete:**

* strategies/momentum.py
* portfolio/construction.py
* portfolio/risk\_metrics.py

**Tasks:**

* Create momentum-based strategy (trend following or cross-sectional).
* Write portfolio construction logic to combine multiple strategies.
* Add support for rebalancing and tracking multiple weights.
* Begin calculating portfolio-level risk metrics.

**GitHub Upload:**

* Upload code and results comparing both strategies individually and combined.
* Push performance summary with updated notebook.

**Week 4: Portfolio Risk Management**

**Goal:** Implement risk controls and improve portfolio analytics.

**Files to complete:**

* portfolio/risk\_management.py
* Update engine.py to support modular constraints
* Optional: reporting/metrics\_summary.py

**Tasks:**

* Add risk constraints: stop-losses, volatility targeting, exposure caps.
* Calculate rolling beta, alpha, and correlation versus benchmark.
* Ensure outputs include annualized volatility and drawdown statistics.

**GitHub Upload:**

* Push code updates and enhanced analytics screenshots.

**Week 5: Simulated Trade Logging and Reporting**

**Goal:** Create a trade log system and structured monthly reports.

**Files to complete:**

* journal/trade\_log.csv or journal/logger.py
* reporting/monthly\_report\_template.html
* Expand generate\_report.py

**Tasks:**

* Log simulated trades with timestamps, PnL, and strategy tags.
* Generate structured monthly reports showing performance summary.
* Include table of trades and performance vs benchmark.

**GitHub Upload:**

* Upload report samples in PDF or image format.
* Include logged trade CSV and analytics output.

**Week 6: Build Streamlit Dashboard MVP**

**Goal:** Create an interactive dashboard to view fund performance.

**Files to complete:**

* dashboard/streamlit\_app.py

**Tasks:**

* Display equity curve, allocation over time, rolling performance.
* Add dropdowns to switch between strategies and timeframes.
* Use Plotly or Seaborn for charts.
* Run locally or deploy on Streamlit Cloud.

**GitHub Upload:**

* Push dashboard code and deployment notes.
* Include dashboard screenshots in README.md.

**Week 7: Final Strategy and Code Cleanup**

**Goal:** Add third strategy and refactor codebase.

**Files to complete:**

* strategies/factor\_investing.py
* config/parameters.yaml
* Final cleanup of engine.py and main.py

**Tasks:**

* Implement a simple factor-based strategy (e.g., value, quality).
* Modularize configuration parameters (weights, lookbacks, etc.).
* Refactor duplicate logic and standardize interfaces across strategies.

**GitHub Upload:**

* Tag as v1.0-beta
* Upload final strategy results and benchmark comparison

**Week 8: Documentation, Demo, and Polish**

**Goal:** Finalize documentation and presentation of the project.

**Files to complete:**

* README.md
* docs/research\_summary.pdf
* Loom video or MP4 walkthrough

**Tasks:**

* Finalize README with project overview, usage instructions, and images.
* Write a 1–2 page research summary explaining strategies and results.
* Record a 3–5 minute demo video showing core features and dashboard.

**GitHub Upload:**

* Tag final version as v1.0
* Include README, documentation, and video link

**Weeks 9–10 (Optional or Stretch Goals)**

Optional polish or stretch goals depending on progress:

* Add machine learning-based signal generation (e.g., Random Forest).
* Integrate alternative data source (news, sentiment, macro data).
* Improve execution simulation with slippage or latency model.
* Deploy dashboard to Streamlit Cloud or Vercel.

**Future Directions**

* Add real-money investment (start small)
* Pitch to student investment clubs or finance groups
* Add real-time data feeds and paper trading API (Alpaca, Interactive Brokers)
* Expand into sector-specific or alternative data strategies
* Create a fund website with investor-facing portal
* Optional future developments: News sentiment, macro indicators, sector rotation

**Optional Enhancements (later on not current worry)**

* Machine Learning-based signal generation
* Macro overlay using economic indicators
* Deploy as cloud app using Heroku, Vercel, or AWS
* Version 2.0: Turn into a real quant startup or hedge fund pitch

**File Organization**

microfund/

├── strategies/

│ ├── momentum.py

│ ├── mean\_reversion.py

│ └── factor\_investing.py

├── backtesting/

│ ├── engine.py

│ └── simulation\_runner.py

├── portfolio/

│ ├── construction.py

│ └── risk\_management.py

├── reporting/

│ ├── monthly\_report\_template.html

│ └── generate\_report.py

├── dashboard/

│ └── streamlit\_app.py

├── data/

│ ├── raw/

│ └── processed/

├── notebooks/

│ └── research\_notebook.ipynb

├── docs/

│ └── research\_notes.pdf

├── config/

│ └── parameters.yaml

├── main.py

├── requirements.txt

└── README.md