# Final Approach

**What this enhanced project adds (why it’s stronger)**

* **Better expected returns** — adds ML (XGBoost / Random Forest) or Bayesian shrinkage for return forecasts instead of naive historical mean.
* **More robust risk** — Ledoit-Wolf shrinkage, factor/PCA or NMF covariance reduction, and CVaR optimization to handle tail risk.
* **Realistic constraints & costs** — explicitly models transaction costs, turnover limits, and min/max weights so results are implementable.
* **Investor personalization** — a fund-recommender module (content + collaborative + demographic + sentiment) chooses candidate funds/assets before optimization, making solutions client-facing and productized. (Your doc’s recommender ideas are directly applied.)
* **Explainability** — LLM (or templated text) generates plain-English explanations for the recommended portfolio and the drivers (top contributing assets, risk exposures).
* **Backtested comparatives** — compare classical Markowitz vs ML-enhanced vs robust optimization on the same historical window (performance, drawdown, turnover).

**High-level architecture (modules / notebook sections)**

1. **Data ingestion** — tickers, funds, and optional news headlines.
2. **Preprocessing & EDA** — returns, missing data, stationarity checks.
3. **Feature engineering / signals** — price-based features (VWAP, OBV slope, ∆OI if available), technicals, sentiment scores. (From your doc: sentiment-aware re-ranking).
4. **Return forecasting (optional ML)** — XGBoost / Random Forest models to predict next-period returns or expected excess returns.
5. **Covariance estimation & dimensionality reduction** — Ledoit-Wolf shrinkage, PCA / factor model or NMF/SVD to stabilize covariance. (Your doc suggests SVD/NMF).
6. **Portfolio construction**
   * Baseline: classical Markowitz mean-variance.
   * Enhanced: ML-forecast + robust covariance + weight constraints.
   * Alternative: CVaR minimization (tail-risk aware).
7. **Fund recommender** (new): content + collaborative + demographic hybrid to pick candidate funds before optimization — then allocate with optimizer.
8. **Backtest & evaluation** — annualized return, volatility, Sharpe, Sortino, max drawdown, turnover, hit rate, information ratio.
9. **Explainability & reporting** — charts, textual explanations (LLM-based templating optional).

**Concrete, feasible data sources (free / public)**

* **Price data (equities / ETFs / funds):** Yahoo Finance via yfinance or pandas\_datareader (free).
* **Mutual fund / ETF metadata:** Yahoo Finance pages, ETFdb, or Kaggle ETF datasets. (Use scraped metadata or manual features.)
* **Economic/factor data (risk-free rate, yields):** FRED (Federal Reserve) via pandas\_datareader.
* **News headlines for sentiment:** NewsAPI (free tier with limits), scraping RSS feeds, or Kaggle news headline datasets. Use only headlines to keep it simple.
* **Options/open interest (if needed):** Yahoo Options pages or CBOE public data (may be limited).
* **If real data is hard or rate-limited:** generate synthetic returns using multivariate normal or via fitted GARCH models; simulate investor preference matrices for collaborative filtering.

**Step-by-step roadmap (ready-to-run MVP checklist)**

Follow this sequence in a single notebook (or modular set of notebooks). I give actionable steps and minimal pseudocode for each step — implementable later.

**Phase A — Setup & baseline (MVP)**

1. **Create project structure**
   * notebooks/01\_data\_and\_EDA.ipynb, 02\_signals\_and\_models.ipynb, 03\_optimization\_and\_backtest.ipynb
2. **Load your current Markowitz notebook** and make the code modular (functions: fetch\_prices(tickers,start,end), compute\_log\_returns(df)).
3. **Extend universe**: increase assets from 2 to ~20 tickers (mix ETFs and stocks) so results are meaningful.
   * Example tickers: SPY, QQQ, IWM, GLD, TLT, VTI, AAPL, MSFT, AMZN, XLF, XLE, XLK, VNQ, EEM
4. **Reproduce baseline**: run your existing random-portfolio simulation and plot return vs volatility. Save baseline metrics.

**Phase B — Robust covariance + eval metrics**

1. **Implement Ledoit-Wolf covariance shrinkage**
   * Use sklearn.covariance.LedoitWolf() to estimate covariance matrix.
2. **Add performance metrics**: Sharpe, Sortino, Max Drawdown, Annualized Return/Volatility, Turnover (if rebalancing).
3. **Add CVaR computation and optimization** (CVaR can be handled with linear programming / cvxpy).
   * Implement a CVaR optimizer for a selected confidence level (e.g., 95%).

**Phase C — Expected returns via ML**

1. **Create features & targets**
   * Features: lagged returns, moving averages, momentum, VWAP proxies (if available), OBV slope.
   * Target: next-month return or next-day return depending on horizon.
2. **Train simple models**
   * Baseline: OLS; ML: RandomForestRegressor and XGBoost (if allowed).
   * Output: predicted expected return vector mu\_ml.
3. **Compare portfolios**

* Solve Markowitz with mu\_hist (historical mean) and mu\_ml.
* Compare performance on hold-out/backtest.

**Phase D — Recommender layer (applies your doc ideas)**

1. **Design fund metadata table** (content features: sector, expense ratio, AUM, risk profile).
2. **Content-based recommendation**: vectorize fund metadata and compute cosine similarity to investor preference vector (risk appetite, sector bias).
3. **Collaborative-style synthetic step**: if you don’t have real investor histories, simulate a small user-item matrix or use open datasets; test SVD/NMF to derive latent preferences. (Your doc lists SVD, NMF.)
4. **Sentiment re-ranking**: get recent news headlines for the top candidate funds/sectors, compute polarity with VADER or TextBlob, and re-rank candidate funds.
5. **Weighted hybrid score**: create score = w1\*content + w2\*collab + w3\*perf + w4\*demographic + w5\*sentiment. Pick top N funds for the optimizer. (Matches your doc hybrid idea.)

**Phase E — Constraints, transaction costs & rebalancing**

1. **Add realistic constraints**:

* Weight bounds: min 0.01, max 0.3
* Turnover penalty: add cost term to objective or run rebalancing frequency monthly/quarterly
* Transaction costs: model as linear cost (bps per trade)

1. **Implement rebalancing & backtest** over rolling windows (e.g., 3-yr lookback, rebalance monthly). Track turnover, costs.

**Phase F — Explainability & comparison report**

1. **Create visual report**: efficient frontier, attribution (which assets contributed to return), factor exposures, heatmaps of covariances.
2. **Add automatic textual explanation**: small LLM prompt (or template) that explains why the optimizer chose specific funds, e.g. “X was overweight because it provided low correlation and high expected excess return from ML signal.” (Your doc suggested LLM evaluation/back-test; use LLM for explanation only.)

**Evaluation plan (what to show on resume / demo)**

* **Backtest table**: Annualized return, Volatility, Sharpe, Sortino, Max Drawdown, Turnover, Net Return after transaction costs — for:
  1. historical-mean Markowitz (baseline),
  2. ML-forecast Markowitz,
  3. Robust (Ledoit+CVAR),
  4. buy-and-hold benchmark (SPY).
* **Visuals**: Efficient frontier overlay (baseline vs enhanced), cumulative returns plot, heatmap of correlation, bar chart of asset allocations.
* **A short demo notebook**: “Click-to-run” cells that produce the final portfolio and explanation text.

**Optional advanced ideas (pick if time permits)**

* **Factor model** (Fama-French factors) to estimate expected returns and risk exposures.
* **Ensemble forecasts**: combine XGBoost + RF + ARIMA using simple weighted average or stacking.