FinLove — Your Money, Our Plan, Your Profit!

Project title: FinLove — Risk-Aware Portfolio Construction
Slogan: Your Money, Our Plan, Your Profit!
Team name: FinLove
Github Link: https://github.com/dinhieufam/FinLove

Title Page

Team members

Full Name	Student ID
Cao Pham Minh Dang	V202401280
Pham Dinh Hieu	V202401287
Nguyen Van Duy Anh	V202401623
Tran Anh Chuong	V202401566
Ngo Dinh Khanh	V202401390

---

1. Introduction

FinLove exists to tackle the pain points of building resilient portfolios in real markets: correlations spike in crises, volatility regimes shift, and naive mean–variance often underestimates tail risk. The project pairs robust risk models, forecast-informed optimizers, and a modern FastAPI + Next.js experience so teams can analyze, predict, and rebalance with realistic data and costs rather than static assumptions.

Key features

- Multiple risk models (Ledoit–Wolf, GLASSO, GARCH, DCC).

- Various optimization methods (Markowitz, Black–Litterman, CVaR, Minimum Variance, Sharpe maximization).
- Realistic backtesting with transaction costs and rebalance bands.
- Forecasting module supporting LSTM, TCN, XGBoost, Transformers, ARIMA, Prophet; SARIMAX selected for production after benchmarking.
- Interactive dashboard with comprehensive visualizations.

---

2. Dataset origin, importance, limitations, and objectives

2.1 Data Source and Collection Methodology

Primary Source: Yahoo Finance via the `yfinance` Python library, providing historical price information dating back several decades for publicly traded securities.

Data Structure: Each ticker symbol is stored as an individual CSV file containing daily OHLCV (Open, High, Low, Close, Volume) records. The data collection process queries Yahoo Finance's API upon user selection, retrieves historical price data, and stores it in a structured format.

Rationale for Selection: Yahoo Finance provides comprehensive market coverage of major U.S. exchanges, offers well-documented Python APIs for easy access, and provides free data for academic research. The historical depth enables robust backtesting and long-term trend analysis.

Data Collection Pipeline: The implementation follows a three-stage process: user-initiated ticker selection, data retrieval with API rate limiting, and intelligent caching with 24-hour expiration to reduce redundant API calls.

2.2 Data Preprocessing and Quality Assurance

Data Cleaning Procedures: Raw financial data often contains inconsistencies that must be addressed before analysis. Our preprocessing pipeline implements several quality control measures. Missing value handling follows a tiered strategy: assets with more than 5% missing observations are automatically excluded, while minor gaps are forward-filled using the last valid price.

Date Alignment and Synchronization: Financial markets operate on different schedules. The preprocessing module aligns all tickers to a common date index, ensuring portfolio-level

calculations use synchronized time periods. Non-trading days are excluded from the aligned dataset.

Return Calculation: Portfolio analysis requires return series rather than raw prices. The system computes daily returns as the percentage change in closing prices from one day to the next. For portfolio-level returns, individual asset returns are aggregated using portfolio weights, where each asset's return is multiplied by its weight and summed across all assets.

Outlier Detection and Treatment: Financial time series occasionally exhibit extreme values due to corporate actions, data errors, or market anomalies. The preprocessing pipeline identifies outliers using statistical methods and flags them for review. In automated processing, extreme returns are capped at the 99th percentile to prevent single-day anomalies from distorting risk estimates.

2.3 Data Limitations and Constraints

Temporal Resolution: The dataset operates at daily frequency, which is adequate for most portfolio management applications but lacks intraday detail. Strategies requiring intraday rebalancing would need additional data sources.

Geographic and Sector Concentration: The implementation is heavily weighted toward U.S. markets, limiting international diversification opportunities. Sector representation may be skewed toward technology and finance sectors.

Data Provider Dependencies: Reliance on `yfinance` introduces operational risks including occasional outages, API changes, and retroactive data revisions. The system implements error handling and fallback mechanisms to mitigate these risks.

Feature Limitations: The dataset is price-centric, excluding fundamental metrics, macroeconomic indicators, alternative data sources, and market microstructure data. This constrains predictive power but keeps the system focused and reproducible.

Data Freshness: The caching mechanism introduces a trade-off between data freshness and efficiency. Users may receive data up to 24 hours old, which is acceptable for portfolio construction but insufficient for real-time trading.

2.4 Project Objectives and Success Criteria

Objective 1: Robust Risk-Aware Portfolio Construction

The primary objective is to provide a portfolio builder that remains robust when market conditions change. Our system addresses regime shifts through shrinkage methods (Ledoit-Wolf), sparse precision estimation (GLASSO), tail-risk-aware optimizers (CVaR), and forecast-informed signals from time-series models.

Objective 2: Analyst-Friendly Web Experience

The second objective focuses on user experience through comprehensive visualizations, intuitive configuration with sensible defaults, LLM-powered explanations, and transparent performance metrics including Sharpe ratios, drawdowns, and transaction costs.

Objective 3: Reproducible Data Pipeline

The third objective ensures reproducibility through versioned data storage with metadata, deterministic processing with fixed random seeds, comprehensive logging, and modular architecture for independent testing.

---

3. Data Science Questions

3.1 Question 1 — Can we use statistical models and company data to produce a superior investment plan?

3.1.1 Introduction

This research investigates whether systematic, algorithm-driven portfolio construction can outperform traditional discretionary investment planning. Quantitative optimization methods leverage mathematical frameworks to systematically balance risk and return objectives, eliminating behavioral biases inherent in intuitive approaches.

Research Objective: Determine whether systematic approaches can produce investment plans that are both theoretically sound and practically superior to discretionary methods, considering risk-adjusted returns, portfolio stability, transaction costs, and robustness to estimation errors.

3.1.2 Approach and Methodology

The optimization framework dynamically selects appropriate methods based on user objectives. Three primary optimization methods form the core:

Classical Mean-Variance Optimization

The Markowitz framework balances expected return against portfolio variance. The optimization maximizes expected utility by finding the optimal trade-off between return and risk, subject to constraints ensuring portfolio weights sum to unity, prohibiting short selling, and limiting maximum position sizes. The method generates efficient frontier curves showing the trade-off between expected return and risk, excelling in stable market conditions but potentially suffering from estimation error during regime shifts.

Bayesian Adjustment via Black-Litterman Model

The Black-Litterman model addresses mean-variance optimization's extreme sensitivity to expected return estimates by combining market equilibrium returns with investor views. The framework computes posterior expected returns that blend market capitalization-based priors with investor confidence-weighted views, producing more stable allocations that anchor to market equilibrium while incorporating specific insights.

Tail Risk Hedging via CVaR Optimization

CVaR optimization explicitly minimizes expected losses in worst-case scenarios, addressing the asymmetry between investor concern for extreme losses versus gains. The method reshapes portfolio allocations to prioritize protection against tail events, underweighting assets with high downside correlation and favoring defensive characteristics. This approach sacrifices some upside potential for improved crisis resilience.

3.1.3 Analysis and Empirical Results

Solver Robustness: The framework automatically selects appropriate solvers (OSQP for quadratic programs, ECOS/SCS for conic programs), ensuring robust operation across different computational environments.

Sensitivity Analysis: Mean-variance optimization is highly sensitive to expected return estimates —a 10% change can shift optimal weights by 20-30%. Black-Litterman addresses this by combining noisy estimates with stable market priors, resulting in 40-50% lower turnover and improved after-cost returns.

Tail Risk Performance: During historical crises (2008, 2020), CVaR-optimized portfolios experienced 15-25% smaller maximum drawdowns compared to mean-variance portfolios. However, during bull markets, CVaR portfolios underperform by 2-4% annually due to defensive positioning.

3.1.4 Discussion and Conclusion

Statistical models can construct superior investment plans when the optimization method matches the context and investor objectives. Sharpe maximization suits growth-oriented investors in stable markets, CVaR optimization benefits conservative investors with strict drawdown constraints, and Black-Litterman harmonizes quantitative optimization with qualitative analyst insights. Success depends on understanding assumptions, limitations, and trade-offs of each approach.

3.2 Question 2 — Can risk models analyze company performance?

3.2.1 Introduction

Traditional performance evaluation focuses on nominal returns, but this provides an incomplete picture. A stock generating 20% returns with extreme volatility may be inferior to one delivering 12% with consistent performance. Performance must be measured as return per unit of risk, where risk encompasses volatility, tail risk, volatility persistence, and systemic exposure.

Research Question: Can sophisticated risk models identify hidden risks, regime-dependent behavior, and systemic vulnerabilities that traditional metrics fail to capture?

3.2.2 Approach — Three-Layered Risk Analysis Framework

Layer 1 — Time-Varying Volatility Modeling (GARCH)

Financial markets exhibit volatility clustering: high volatility periods tend to persist. GARCH models capture this by modeling volatility as time-varying, with a persistence parameter indicating how long volatility shocks persist. Assets with high persistence remain volatile for extended periods once a high-volatility regime begins, while those with low persistence quickly revert to normal levels. This temporal dimension is invisible to static metrics like Sharpe ratios.

Layer 2 — Structural Correlation Analysis (Ledoit-Wolf and GLASSO)

Ledoit-Wolf shrinkage provides robust covariance estimation by shrinking sample covariance toward a structured target, improving out-of-sample performance. GLASSO estimates sparse precision matrices, revealing conditional independence relationships and identifying "hub" assets —securities that are conditionally dependent on many others. Banking stocks frequently emerge as hubs, meaning shocks propagate widely through the system, as demonstrated during the 2008 financial crisis.

Layer 3 — Tail Risk and Drawdown Metrics

Maximum drawdown measures the largest peak-to-trough decline, answering what the worst loss would have been. CVaR (Conditional Value at Risk) quantifies expected losses in worst-case scenarios, providing more informative tail risk measures than VaR by considering the magnitude of losses beyond the threshold.

3.2.3 Analysis — Key Empirical Findings

Hidden Risk Discovery: Some high-growth technology stocks exhibit moderate volatility (15-20%) but extremely large CVaR values (losses exceeding 30% in worst scenarios), indicating fat-tailed distributions. Traditional metrics miss this hidden risk.

Regime-Dependent Volatility: Technology stocks show high volatility persistence (0.85-0.95), remaining volatile for extended periods after spikes, while utilities show lower persistence (0.70-0.80). During stable conditions, both may show similar Sharpe ratios, but GARCH reveals technology stocks are more vulnerable to regime shifts.

Systemic Exposure: GLASSO analysis reveals banking stocks as central hubs in the financial network. Portfolios with high banking exposure experienced cascading losses during 2008, demonstrating the importance of understanding systemic dependencies.

3.2.4 Discussion and Conclusion

The three-layered framework demonstrates that risk is multidimensional: temporal (volatility clustering), structural (correlation networks), and tail-related (extreme events). Combining insights yields comprehensive risk profiles enabling informed trade-offs. Quality—defined as risk-adjusted stability and resilience—often trumps raw return magnitude, making this approach essential for long-term wealth preservation.

3.3 Question 3 — Can we predict cumulative return and volatility?

3.3.1 Introduction

Forecasting portfolio returns and volatility enables proactive portfolio management, dynamic hedging, and stress testing. However, financial time series exhibit autocorrelation, seasonality, and high noise-to-signal ratios that make prediction extremely difficult.

Research Objective: Determine whether modern forecasting techniques can produce reliable forecasts of portfolio returns and volatility, considering both point forecasts and uncertainty quantification.

3.3.2 Dataset Features and Preprocessing

Primary Data Sources: Daily closing prices and portfolio-level returns computed as weighted averages of individual asset returns, where each asset's return is multiplied by its portfolio weight and summed. Portfolio weights are derived from chosen optimization methods, ensuring forecasts reflect actual portfolio composition. The dataset is divided into training (70%), validation (15%), and test (15%) sets using temporal splitting.

Design Decision: The analysis excludes exogenous macroeconomic variables to focus on the intrinsic predictive power of return series patterns, maintaining experimental reproducibility.

3.3.3 Approach — Comprehensive Model Benchmarking

Model Selection: Five model classes were evaluated: ARIMA (classical linear model), Prophet (additive decomposition), LSTM (recurrent neural network), TCN (temporal convolutional network), and Transformer (attention-based architecture).

Evaluation Metrics: MAE (mean absolute error), RMSE (root mean squared error), MAPE (mean absolute percentage error), and R^2 (coefficient of determination).

Benchmark Results:

Model	MAE	RMSE	MAPE	R^2
ARIMA	0.007462	0.009559	125.31	-0.091
Prophet	0.007578	0.009738	127.64	-0.133
LSTM	0.007690	0.009772	154.28	-0.153
TCN	0.007637	0.009656	153.68	-0.121
Transformer	0.008844	0.011024	402.08	-0.586

Key Finding: ARIMA/SARIMAX consistently achieved the lowest MAE and RMSE. Deep learning models underperformed despite theoretical capacity, highlighting that model complexity

only helps when sufficient signal exists. Financial returns, with high noise-to-signal ratios, favor simpler, regularized models.

3.3.4 Why ARIMA/SARIMAX Emerged as Optimal

ARIMA/SARIMAX offers several advantages: lower sensitivity to noise (3-7 parameters vs. millions in neural networks), faster training (seconds vs. hours), interpretable residuals for diagnostics, and better short-term stability capturing momentum and mean reversion patterns. The seasonal SARIMAX extension captures weekly patterns and day-of-week effects (Monday effects, end-of-week effects) common in financial markets.

Hyperparameter Search: Systematic search within constrained ranges (autoregressive and moving average orders 0-2, seasonal period 5 or 7) using AIC minimization, with computational constraints (20 seconds max tuning time, 100 max iterations).

3.3.5 Volatility and Cumulative Return Prediction

Volatility Forecasting: Uses a two-stage process: SARIMAX generates daily return forecasts, then annualized volatility is computed as the standard deviation of forecast returns multiplied by the square root of trading days per year. Residual-based noise injection addresses the issue of unrealistically smooth forecasts by adding noise from historical residuals to restore realistic volatility levels.

Cumulative Return Forecasting: Historical cumulative returns are computed by multiplying one plus each daily return sequentially, ensuring proper compounding. Forecast cumulative returns extend the historical series by multiplying the last historical cumulative value by the product of one plus each forecasted return. However, forecast uncertainty compounds with horizon, making cumulative return forecasts increasingly unreliable for long-term planning.

3.3.6 Analysis and Discussion

ARIMA/SARIMAX delivered the best accuracy by capturing dominant patterns (short-term autocorrelation, mean reversion) while avoiding overfitting. Negative R^2 values across all models reflect the fundamental difficulty of predicting financial returns, but ARIMA's relative superiority suggests it extracts maximum available signal.

Volatility forecasts are substantially more stable and reliable than cumulative return forecasts, as volatility is less sensitive to forecast errors and exhibits stronger persistence. Limitations include: no exogenous macro features limiting accuracy during economic shifts, limited long-horizon predictive power beyond 10-15 days, underreaction to sudden regime shifts, and point forecasts

only (no distributional forecasts). Despite limitations, the framework provides valuable forward-looking information for dynamic portfolio construction.

---

4. Conclusion

4.1 Main insights

- Financial time series are noisy and structurally unstable; model complexity only matters when data supports it. Classical models (SARIMAX) can outperform deep architectures in low-signal daily return contexts due to stability and interpretability.
- Data pipeline hygiene (caching, versioning, reproducible downloads) is essential to reduce latency and ensure consistent EDA and backtest results.
- The most informative features for risk estimation and optimization were returns, volatility, momentum, and covariance structure. These guided the end-to-end workflow from signal extraction to optimized allocations.
- Productization and UX matter: integrating LLM-driven assistance and simplified configuration reduced user friction and increased platform accessibility.

4.2 Potential extensions

- Modeling: regime-switching models, volatility-targeting frameworks, or richer exogenous macro and alternative data sources.
- Optimization: robust optimization methods, risk-parity, or reinforcement learning for allocation strategies.
- Infrastructure: containerized, cloud-native deployment, asynchronous task queues, scheduled retraining, and real-time ingestion.
- User experience: dynamic onboarding with risk profiling, scenario simulations, and real-time conversational feedback from the LLM assistant.

---

5. Lifecycle Reflection

Demonstrates adherence to the Data Science Life Cycle phases:

Problem & business understanding

The project aims to address portfolio fragility during market stress—specifically tackling spiking correlations, volatility clustering, and "fat tail" events. The solution provides Portfolio Managers (PMs) and analysts with risk-aware allocation strategies, forward-looking return forecasts, and realistic constraints (transaction costs, latency) to ensure robust decision-making in volatile environments.

Data collection

Pull daily OHLCV (Open, High, Low, Close, Volume) records from Yahoo Finance via `yfinance`, with pre-download and 24-hour caching in `data_cache/` and project CSVs under `data/` folder for reproducibility.

Data processing

- Clean multi-index price files, align dates, drop assets with heavy missingness, compute returns/log-returns, and persist artifacts.

Data analysis

We conduct a comprehensive statistical analysis to validate model assumptions. This includes exploring trends in the price of each ticker, volumes, return distributions, rolling correlations/ volatility, and cross-asset relationships (heatmaps, scatter/volatility plots) to surface liquidity, regime shifts, and diversification limits.

Modeling

We apply a variety of models ranging in different purposes. They include:

- Risk: Ledoit–Wolf shrinkage, GLASSO.
- Volatility: GARCH/DCC.
- Optimization: Markowitz, Min-Var, Sharpe, Black–Litterman, CVaR.
- Forecasting: ARIMA/Prophet and neural models; SARIMAX chosen for production.

Deployment

Serve models and data through FastAPI (`web/backend`) with prediction and portfolio endpoints; deliver the UI via Next.js (`web/frontend`); reuse the same cached data pipeline so EDA findings match the live app and backtests. For development, testing, and demonstration

purposes, the application utilizes ngrok to create secure tunnels to the localhost environment, allowing the locally deployed services to be accessed externally.

---

6. Team Contribution Statement

Name	Contribution	Status
Tran Anh Chuong	Lead data preprocessing and exploratory data analysis; design and implementation of analytics UI.	Done
Cao Pham Minh Dang	Development, training, and validation of prediction models and forecasting pipeline.	Done
Pham Dinh Hieu	LLM integration, RAG pipeline, advanced dashboard intelligence features and UX improvements.	Done
Ngo Dinh Khanh	Front-end engineering, dashboard UX and landing page design/ implementation.	Done
Nguyen Van Duy Anh	Development of quantitative risk-assessment models and optimization algorithms.	Done

---

7. Individual Reflections

Tran Anh Chuong

The task I was assigned is related to Data & Infrastructure. Acquire via yfinance, caching & versioning; build feature store. Serving as data engineer and analyst in the project, I was responsible for constructing a pipeline to collect data, though it seems simple at first when there is the yfinance library for crawling historical data for tickers in Yahoo Finance, the most critical part is on the production stage where fetching data of targeted ticker chosen by user and caching them is crucial to ensure that the server don’t store those .csv files permanently, reducing the latency for the whole process. For the data analyst role, I didn’t just use a basic exploratory data analysis (EDA) strategy generally, but I had to get a sense of financial domain knowledge, which was relatively challenging for me since I have almost zero background in finance. Through exploring essential features such as returns, momentum, vol, Σ, etc. Not only did I overcome the

fear in finance facts, but I was also able to conduct a comprehensive EDA notebook that can deliver core insights from several tickers, such as closing price trend, daily average returns, rolling volatility volume snapshot, etc. FinLove has undoubtedly been a great opportunity for me to practice my data science skills in data engineering and analytics, especially in a new domain that I have never experienced before.

Cao Pham Minh Dang

In this project, my main contribution was designing, training, and evaluating the forecasting models. Working through this process taught me far more than I expected. I initially assumed that deep learning models would dominate our results, but comparing them directly with classical approaches challenged that belief. Financial return series turned out to be noisier and more fragile than I had imagined, and models such as LSTM, TCN, and Transformers struggled to extract meaningful structure. In contrast, simpler statistical methods often produced more stable and accurate predictions.

Selecting and tuning the SARIMAX model became a turning point in my understanding. Instead of relying on architectural complexity, I had to focus on seasonal patterns, autoregressive behavior, and proper residual diagnostics. This forced me to slow down, test assumptions, and pay attention to the underlying time-series properties that I would have overlooked before. The experience made it clear how quickly deep learning can overfit when the data does not support high model capacity.

Through this work, I learned to value evidence over intuition and robustness over novelty. The process strengthened my technical discipline and gave me a clearer sense of how forecasting models behave in real financial environments. It also reminded me that the most reliable model is not the most sophisticated one, but the one that continues to perform when confronted with real data.

Pham Dinh Hieu

In this project, I was responsible for the LLM integration, the RAG system, and several dashboard features. More broadly, I focused on the user-experience aspect of the website. One of my main tasks was improving the interface to make it more accessible for non-expert users. For example, our initial configuration page required users to manually select a model name, which was confusing and too technical. I redesigned this into a simple multiple-choice flow based on user characteristics, which then mapped to the appropriate risk model and optimisation algorithm. Similarly, instead of letting users choose prediction models directly, I adjusted the system to

automatically use the best-performing model determined during evaluation.

The most interesting feature I implemented was the LLM assistant. When testing the website, I realised that some of the strategy explanations were difficult to understand, so we integrated an interactive LLM to help users interpret the analysis. The main challenge here was performance. HuggingFace models were accurate but too slow for real-time Q&A. To solve this, I used a lightweight embedding model for RAG and switched the Q&A component to the Gemini API, which provided much faster responses.

Through this project, I strengthened my skills in web development, fintech concepts, and model deployment, as well as user-centred design. Working in a team also taught me the importance of clear task allocation. In future projects, I hope we can improve our GitHub workflow so our branches and updates are more organised.

Ngo Dinh Khanh

My primary focus in this project was full-stack web development for the FinLove application, a role that required me to act as the bridge between complex data science and the end-user. One of my biggest hurdles was the sheer density of financial statistics; initially, I struggled to grasp the correlations between various portfolio metrics. However, by navigating with curiosity and collaborating closely with my teammates, I not only strengthened my knowledge of portfolio analysis but also learned to put myself in the user's shoes—translating these rigid technical processes into intuitive support for new investors. On the technical side, engineering the pipeline between our self-trained FastAPI models and the Next.js frontend was a steep learning curve, as it was my first time integrating local AI architectures with a modern web framework. I also moved beyond standard development by mastering deployment techniques like tunneling with ngrok, which allowed us to transform FinLove from a mere local demo into a true, accessible application. This journey has significantly matured my ability to build software that is not just functional, but understandable and usable.

Nguyen Van Duy Anh

My contribution to the FinLove project centered on architecting a rigorous statistical framework that transforms raw financial data into mathematically optimal investment strategies. I developed the risk module to solve the "garbage in, garbage out" problem by implementing robust covariance estimators such as Ledoit-Wolf shrinkage and Graphical Lasso, which stabilize inputs against market noise, along with GARCH models for capturing time-varying volatility. From here, I engineered the optimize module using cvxpy to solve complex convex optimization

problems, implementing advanced objectives like Black-Litterman, which integrates Bayesian views to prevent overfitting, and Conditional Value at Risk, CVaR, which explicitly minimizes tail losses via Linear Programming. To validate these models, I designed a comprehensive metrics suite that audits performance beyond simple returns, quantifying "hidden" factors like turnover costs and weight stability to ensure strategies are not just theoretically sound, but practically executable. Apart from that, I have also learned a lot of interesting finance domain knowledge through this project regarding how risks become highly important when seeing a potential portfolio. Mathematics and Probability & Statistics also become essential for me to acquire all the knowledge in building statistical models and developing optimization functions for FinLove, that is such a perfect opportunity to apply what I have learned into a real-world problem.

---

8. References

1. Teplova, T., Evgeniia, M., Munir, Q., & Pivnitskaya, N. (2022). *Black–Litterman model with copula-based views in a mean-CVaR portfolio optimization framework with weight constraints.* Economic Change and Restructuring, 56(1), 515–535. https://doi.org/10.1007/s10644-022-09435-y
2. El Karoui, N., Lim, A. E. B., & Vahn, G.-Y. (2012). *Performance-based regularization in mean-CVaR portfolio optimization* (Version 2). arXiv. https://doi.org/10.48550/arXiv.1111.2091
3. Zhou, Q. (2024). *Portfolio optimization with robust covariance and Conditional Value-at-Risk constraints* (Version 1). arXiv. https://doi.org/10.48550/arXiv.2406.00610
4. Millington, T., & Niranjan, M. (2017). *Robust portfolio risk minimization using the graphical lasso.* In D. Zhao et al. (Eds.), Neural Information Processing – ICONIP 2017 (Lecture Notes in Computer Science, vol. 10635) (pp. 863–872). Springer. https://doi.org/10.1007/978-3-319-70096-0_88

---

9. Bonus

9.1 LLM Integration

The platform includes an LLM layer via a Retrieval-Augmented Generation (RAG) pipeline to ground explanations in portfolio data, risk metrics, and forecasts. The module supports multiple providers (GPT family, Gemini) and rotates API keys to manage rate limits. The LLM provides interpretative assistance and educational explanations—not investment advice—and focuses on improving user understanding of model outputs and visualizations.

9.2 Website deployment

Architecture: Next.js frontend & FastAPI backend. Backend hosts portfolio engines, risk models, optimization, forecasting pipelines, and RAG. Frontend proxies API requests to the backend (frontend: port 3000; backend: port 8000). Deployment options: local development, cloud hosting, or containerized production with CI/CD. Recommended production improvements include containerization, task queues for asynchronous computation, scheduled retraining pipelines, and feature-store extensions for cross-asset analytics.

End of report.