TradeGenie

Mohammed Mudassiruddin

PROBLEM STATEMENT

Financial traders today struggle with fragmented data sources and siloed analytics platforms that hinder timely decision-making and obscure critical market insights.

The proliferation of diverse instruments from equities and derivatives to algorithmic high-frequency trading exacerbates operational complexity and elevates systemic risk, demanding sophisticated AI oversight to maintain stability and performance.

Therefore, a pressing need for an end-to-end, modular platform that seamlessly orchestrates data ingestion, multi-agent strategy execution, risk management, back testing, and real-time visualization

SOLUTION

TradeGenie offers a streamlined, user-friendly solution that empowers traders to visualize market insights, generate data-driven signals, and manage portfolio risk through an intuitive web interface and a lightweight backend—all without requiring deep technical know-how.

Users can simply install the application, configure their API keys, and begin exploring prebuilt trading strategies, backtesting scenarios, and real-time analytics in minutes.

This approach makes TradeGenie ideal for investors seeking an AI-powered toolkit to enhance decision-making and learn algorithmic trading workflows, while abstracting away the complexity of machine-learning models, containerization, or infrastructure setup.

TECH STACK

TradeGenie uses a modern, full-stack setup built for speed and simplicity:

The frontend is powered by React and Vite for a snappy, component-driven user interface, while the backend is a lightweight Python service built with FastAPI and managed via Poetry, served on Uvicorn for fast, asynchronous request handling.

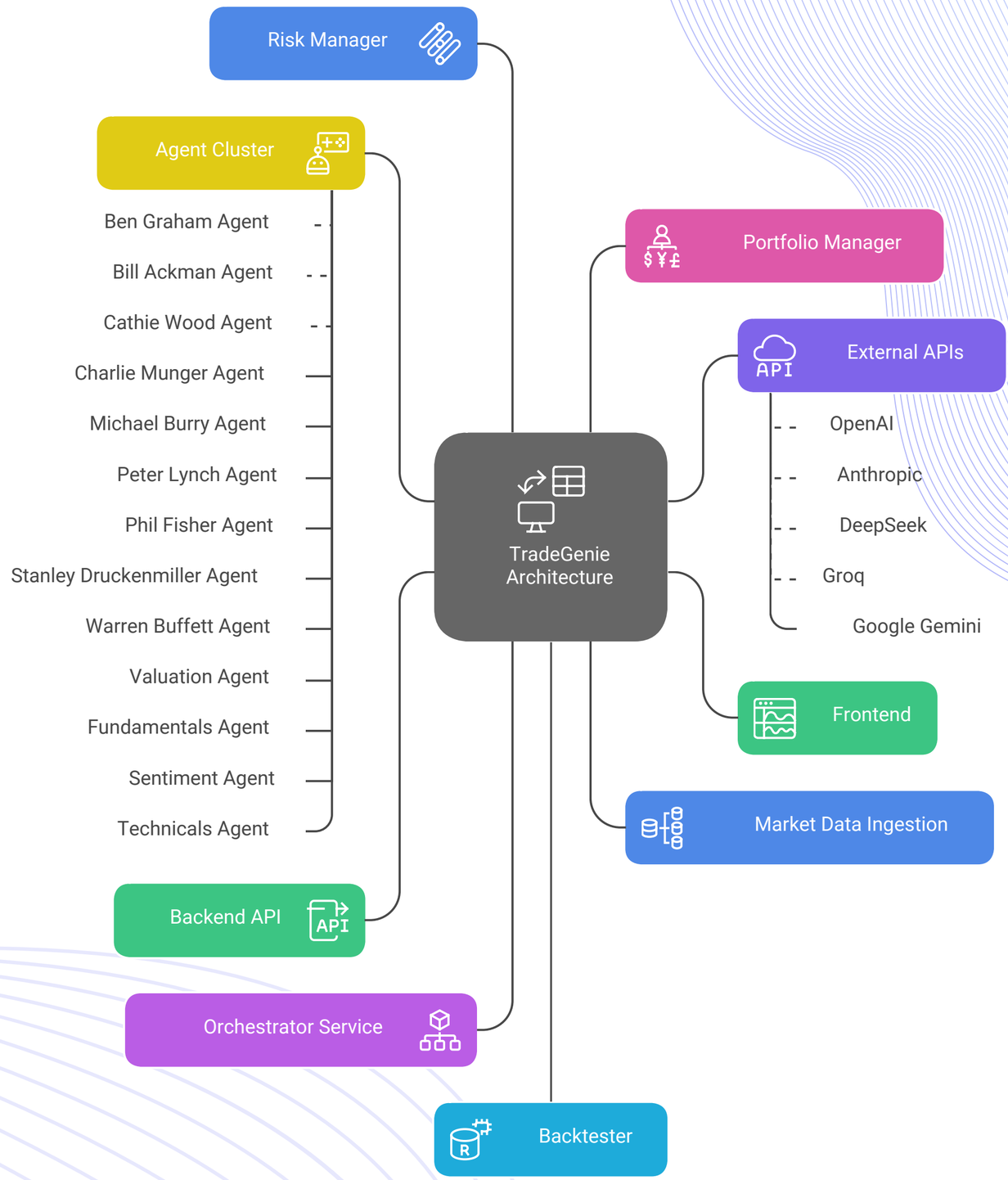
State and data flow are handled through familiar patterns in both React and Python, and all configuration (API keys, environment settings) is managed via simple .env files.

Pydantic ensures type-validated settings and API schemas, minimizing runtime surprises. Instead of complex container orchestration, the app runs seamlessly with just a few CLI commands making local development, staging, and deployment as frictionless as possible. We also use Prettier, ESLint, and black to maintain consistent code formatting and linting across the stack.

USE CASES

- Rapidly prototype and backtest AI-driven trading strategies on historical market data.
- Generate real-time buy/sell signals using combined technical, fundamental, and sentiment agents.
- Automatically evaluate portfolio risk metrics and enforce dynamic position limits.
- Optimize multi-asset allocations by aggregating insights from value, growth, and contrarian models.
- Visualize strategy performance and signal analytics through an interactive React dashboard.
- Compare the impact of different LLM-based trading “agents” side-by-side for research.
- Serve as an educational sandbox for learning AI-powered market analysis and backtesting.

ARCHITECTURE



THANK YOU