Product Requirements Document (PRD)

Product Name: Quant Trading Engine MVP **Owner**: [Your Name] **Date**: June 2025 **Version**: 1.0

---

1. Objective

To build a modular, scalable, and efficient Quantitative Trading Engine that supports data ingestion, strategy development, backtesting, paper/live trading, and risk management. This engine will allow rapid prototyping and deployment of trading strategies across multiple markets and brokers.

---

2. Target Users

- Quantitative Traders
- Data Scientists
- Trading Strategy Developers
- Fintech Startups

---

3. Features & Requirements

3.1 Data Ingestion

- Ingest historical OHLCV data (CSV/API).
- Real-time price feed integration (WebSockets/REST).
- Support for multiple timeframes: 1m, 5m, 1h, daily.

3.2 Strategy Engine

- Plug-and-play architecture for custom strategies.
- Strategy inputs: price, indicators, ML signals.
- Signal output: BUY, SELL, HOLD.
- Support for multiple instruments & timeframes.

3.3 Backtesting

- Historical data simulation engine.
- Metrics: Sharpe ratio, Max Drawdown, Win Rate, CAGR.
- Configurable slippage, commission, execution delays.

3.4 Paper Trading

- Simulated order execution.
- Tracks virtual positions, PnL, and logs.
- Realtime data driven but risk-free.

3.5 Live Trading

- Broker integration (Zerodha, Alpaca, IBKR).
- Order types: Market, Limit.
- Live position tracking, order acknowledgement.
- Session recovery on restart.

3.6 Risk Management

- Per-trade SL/TP configuration.
- Portfolio exposure limits.
- Circuit breakers for session loss control.
- Daily capital usage constraints.

3.7 Portfolio Management

- Real-time cash, holdings, PnL tracker.
- Leverage & margin tracking.
- Multi-asset support (equities, futures, crypto).

3.8 Reporting & Visualization

- Live dashboard (PnL, performance, logs).
- Equity curve, drawdown chart, exposure analytics.
- Export reports in CSV/JSON.

3.9 Configuration & Extensibility

- YAML/JSON based configuration for strategies and parameters.
- CLI & API interfaces for automation.

---

4. Non-Functional Requirements

4.1 Performance

- Backtest speed: <10 sec for 1Y minute-level data.
- Real-time latency: <300ms for trade signal to execution.

4.2 Scalability

- Handle 100+ concurrent instruments.
- Cloud-ready: Docker + Kubernetes compatible.

4.3 Reliability

- Auto-retry on data/API failures.
- Resilient session restore from disk (SQLite/PostgreSQL).

4.4 Security

- Encrypted API key storage.
- Role-based access control (if web UI is used).

4.5 Maintainability

- Modular codebase with unit/integration tests.
- Logging at multiple levels with rotating logs.

---

5. Milestones & Timeline

Phase	Deliverables	Timeframe
Phase 1	Data ingestion, basic strategy engine, backtesting	Week 1-2
Phase 2	Paper trading, risk module, logging	Week 3-4
Phase 3	Live trading integration, dashboards	Week 5-6
Phase 4	Portfolio management, documentation	Week 7

---

6. Dependencies

- Broker APIs (Zerodha, Alpaca)
- Market data APIs (Yahoo, Binance, Polygon.io)
- Python libraries: pandas, numpy, ta-lib, Flask, websockets

---

7. Success Metrics

- <300ms latency from signal to execution
- 90%+ uptime during market hours
- Strategy plug-in in <15 minutes
- Backtest speed > 1 year/minute-level in <10s

---

8. Future Scope (Phase 2+)

- Machine learning model integration
- Telegram/Slack alert bots
- Options trading support
- Portfolio optimization module