**Interactive Full-Stack Simulated Trading Platform**

**Overview**

A website application featuring simulated stock market trading that allows users to enter an arbitrary amount of money which will then be simulated using various trading algorithms in an attempt to increase the starting funds over a user-specified time frame (days, months, years). The platform will display comprehensive results, statistics on performance, and ROI metrics using historical market data with realistic trading conditions.

**Tech Stack:**

**Frontend:**

* React with TypeScript for type safety
* React Query/Redux for state management
* Tailwind CSS for styling
* Chart.js or Recharts for data visualization

**Backend:**

* C++ for core algorithmic processing and performance-critical calculations
* FastAPI (Python) as the bridge between C++ backend and web frontend

**Data Sources:**

* Primary: Polygon.io (comprehensive historical data)
* Alternative: Alpaca API or Yahoo Finance
* Local data preprocessing and storage for optimal performance

**Database:**

* PostgreSQL for time-series historical market data (with TimescaleDB extension)
* Redis for caching frequently accessed data and session management

**Market Data**

* **Historical Range:** 1980-2025 (or available range from data provider)
* **Data Points:** OHLCV (Open, High, Low, Close, Volume) at various intervals
* **Storage:** Preprocessed and stored locally with efficient indexing
* **Access Control:** Algorithms only access data up to simulation's "current" date (no look-ahead bias)
* **Asset Coverage:** Stocks, ETFs, and major indices for benchmarking

**Trading Algorithms**

**Phase 1 - Basic Strategies**

1. **Moving Average Crossover**
   * Simple MA and Exponential MA variants
   * Configurable periods (e.g., 50/200 day)
2. **Mean Reversion**
   * Bollinger Bands strategy
   * RSI oversold/overbought levels
3. **Momentum Trading**
   * RSI-based entries
   * MACD signal crossovers

**Phase 2 - Advanced Strategies**

1. **Statistical Arbitrage / Pairs Trading**
   * Cointegration tests
   * Z-Score trading
   * Kalman Filters for dynamic hedge ratios
2. **Multi-Factor Models**
   * Combining technical indicators
   * Risk-adjusted position sizing
3. **Bayesian Methods**
   * Dynamic parameter optimization
   * Regime detection

**Risk Management Components**

* **Position Sizing:** Kelly Criterion, Fixed Fractional, or Equal Weight
* **Stop-Loss Mechanisms:** Trailing stops, fixed percentage, ATR-based
* **Portfolio Constraints:**
  + Maximum position size per asset
  + Sector diversification limits
  + Maximum portfolio leverage
* **Risk Metrics:**
  + Maximum drawdown limits
  + Value at Risk (VaR) calculations
  + Real-time exposure monitoring

**Realistic Simulation Factors**

* **Transaction Costs:**
  + Configurable commission structure
  + Bid-ask spread modeling
  + Slippage based on order size and liquidity
* **Market Mechanics:**
  + Order types (market, limit, stop)
  + Partial fills for large orders
  + Market hours and trading halts
* **Capital Constraints:**
  + Margin requirements
  + Pattern day trader rules (if applicable)
  + Cash settlement periods

**Performance Metrics & Analytics**

**Core Metrics:**

* Total Return & Annualized Return
* Sharpe Ratio & Sortino Ratio
* Maximum Drawdown & Recovery Time
* Win Rate & Profit Factor
* Risk-Adjusted Return (Information Ratio)

**Visualization:**

* Equity curve with drawdown periods
* Trade distribution histogram
* Monthly/Yearly return heatmaps
* Benchmark comparison charts (vs S&P 500)
* Detailed trade log with entry/exit points

**Implementation Roadmap**

**Phase 1: Foundation (Months 1-2)**

* Set up database schema and data pipeline
* Implement core simulation engine in C++
* Create FastAPI bridge with basic endpoints
* Build simple React frontend with basic charts
* Implement first trading algorithm (MA Crossover)

**Phase 2: Core Features (Months 3-4)**

* Add transaction cost modeling
* Implement 2-3 additional basic strategies
* Create comprehensive performance analytics
* Add portfolio-level risk management
* Develop trade visualization components

**Phase 3: Advanced Features (Months 5-6)**

* Implement advanced trading strategies
* Add Monte Carlo analysis for strategy robustness
* Create strategy comparison tools
* Implement real-time simulation progress
* Add export functionality for results

**Phase 4: Polish & Optimization (Month 7+)**

* Performance optimization for large-scale backtests
* Advanced caching strategies
* UI/UX improvements
* Strategy parameter optimization tools
* Community features (strategy sharing, leaderboards)

**Additional Considerations**

* **Data Quality:** Implement checks for splits, dividends, and corporate actions
* **Scalability:** Design for parallel processing of multiple simulations
* **User Education:** Include tooltips and documentation for financial metrics
* **Legal:** Clear disclaimers about simulated vs real trading