**INTERNSHIP REPORT**

Quantitative Momentum Backtesting System

Student Name: [Your Name]  
Internship Duration: [Start Date] - [End Date]  
Organization: [Company Name]  
Supervisor: [Supervisor Name]

# 1. EXECUTIVE SUMMARY

This report presents the development and implementation of a comprehensive quantitative momentum backtesting system. The project involved designing and building a production-grade financial analysis tool capable of evaluating momentum-based trading strategies across a universe of 119 stocks over a 9+ year period (June 2016 - August 2025). The system successfully processes over 298,000 data points and executes thousands of simulated trades to analyze strategy performance with institutional-grade metrics and visualizations.

# 2. PROJECT OBJECTIVES

The primary objectives of this project were:  
  
â€¢ Develop a robust backtesting engine for momentum-based trading strategies  
â€¢ Implement accurate performance metrics (CAGR, Sharpe Ratio, Maximum Drawdown, Hit Ratio)  
â€¢ Create an interactive web-based user interface for strategy analysis  
â€¢ Enable comparative analysis across different price types (adjusted vs. raw closing prices)  
â€¢ Provide detailed quarterly performance tracking and portfolio selection insights  
â€¢ Ensure scalability and accuracy for institutional-grade financial analysis

# 3. METHODOLOGY & TECHNICAL IMPLEMENTATION

## 3.1 System Architecture

The system was built using a modular architecture with clear separation of concerns:

â€¢ Core Engine (src/core/engine.py): Executes backtest simulations with daily mark-to-market calculations  
â€¢ Data Loader (src/core/data\_loader.py): Handles CSV parsing, date format detection, and price matrix construction  
â€¢ Analytics Module (src/core/analytics.py): Computes CAGR, Sharpe ratio, drawdown, and quarterly metrics  
â€¢ Strategy Module (src/core/strategy.py): Implements momentum ranking and portfolio selection logic  
â€¢ Visualization Layer (src/core/plotting.py): Creates interactive charts using Plotly  
â€¢ Web Interface (src/app/streamlit\_app.py): Streamlit-based UI for user interaction

## 3.2 Trading Strategy Implementation

The momentum strategy operates as follows:  
  
â€¢ Universe: 119 stocks with daily price data from October 2015 to August 2025  
â€¢ Lookback Period: 3 months for momentum calculation  
â€¢ Selection Criteria: Top 24 stocks ranked by highest 3-month returns  
â€¢ Rebalancing Frequency: Quarterly (March 31, June 30, September 30, December 31)  
â€¢ Portfolio Allocation: Equal-weight allocation across selected stocks  
â€¢ Cost Model: Transaction costs (10 bps) and market impact slippage (5 bps)  
â€¢ Starting Capital: $1,000,000

## 3.3 Technical Challenges & Solutions

**â€¢** Date Format Handling: Implemented flexible date parsing supporting multiple formats (DD-MM-YYYY, YYYY-MM-DD, Excel serial dates)  
**â€¢** Position Holding Bug: Fixed critical issue where portfolio positions were incorrectly liquidated between rebalances  
**â€¢** Cash Accounting: Corrected double-counting of equity by properly tracking cash vs. invested capital  
**â€¢** Quarterly Returns Calculation: Redesigned to calculate from first-to-last business day of each quarter rather than compounding daily returns  
**â€¢** Cache Management: Implemented session state clearing to prevent stale results when parameters change  
**â€¢** Performance Optimization: Optimized data structures for handling 2,500+ days Ã— 119 tickers efficiently

# 4. RESULTS & KEY FINDINGS

The backtesting system successfully generated comprehensive performance analytics:  
  
â€¢ Backtest Period: June 30, 2016 - August 31, 2025 (9.2 years)  
â€¢ Total Return: 76.5% (using adjusted close prices)  
â€¢ Compound Annual Growth Rate (CAGR): 6.43%  
â€¢ Sharpe Ratio: 0.36 (indicating moderate risk-adjusted returns)  
â€¢ Maximum Drawdown: -33.98% (occurred during COVID-19 crash in March 2020)  
â€¢ Quarterly Hit Ratio: 38.5% (15 out of 39 quarters positive)  
â€¢ Total Trades Executed: 1,800+ transactions over 39 rebalancing periods  
â€¢ Average Momentum Score: Top selected stocks showed 15-25% returns over lookback period

## 4.1 Key Insights

â€¢ Price Type Impact: Using adjusted close prices yielded 16% higher returns than raw close prices (76.5% vs. 60.4%), highlighting the importance of corporate action adjustments  
â€¢ Volatility Events: The strategy experienced significant drawdown during the 2020 pandemic but recovered over subsequent quarters  
â€¢ Momentum Persistence: Selected stocks demonstrated varying performance, with ticker-level hit ratios ranging from 35-45%  
â€¢ Quarterly Patterns: Portfolio value showed clear quarterly inflection points corresponding to rebalancing dates  
â€¢ Transaction Costs: Total fees and slippage reduced returns by approximately 150 basis points annually

# 5. TECHNICAL DELIVERABLES

â€¢ Core Backtesting Engine: 9 Python modules with 2,000+ lines of production code  
â€¢ Test Suite: 70+ unit tests ensuring code reliability and accuracy  
â€¢ Interactive Web Application: Streamlit-based interface with real-time visualizations  
â€¢ Performance Visualizations: Equity curve, drawdown chart, rolling Sharpe ratio, quarterly heatmap  
â€¢ Data Export Capabilities: CSV downloads for trades, holdings, returns, and quarterly selections  
â€¢ Comprehensive Documentation: Code documentation, README, and deployment guides  
â€¢ Version Control: Git repository with 25+ commits tracking development progress

# 6. SKILLS & TECHNOLOGIES UTILIZED

**Programming & Libraries:** Python 3.13, Pandas, NumPy, Plotly, Streamlit  
**Financial Concepts:** Momentum investing, backtesting methodology, risk metrics, portfolio management  
**Software Engineering:** Object-oriented design, modular architecture, unit testing, version control (Git/GitHub)  
**Data Processing:** Time series analysis, data validation, matrix operations, date parsing  
**Web Development:** Interactive UI design, caching strategies, session state management

# 7. CONCLUSION & LEARNING OUTCOMES

This internship project provided invaluable hands-on experience in quantitative finance and software development. The successful implementation of a production-grade backtesting system demonstrated the practical application of financial theory, data science, and software engineering principles. Key learnings include the critical importance of accurate data handling, the nuances of portfolio accounting, and the value of comprehensive testing in financial applications.  
  
The project reinforced the understanding that seemingly small bugs (such as the cash accounting error) can have dramatic impacts on calculated returns, emphasizing the need for rigorous validation. Additionally, the experience of building an end-to-end systemâ€”from data ingestion to user interfaceâ€”provided insights into the full software development lifecycle in a financial context.  
  
Moving forward, the system can be extended with additional strategy types, risk management features, and real-time data integration, providing a solid foundation for future quantitative research and analysis.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Student Signature  
Date: October 28, 2025