International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 14, Issue 2, 2026

Agentic Trading Bot

Dr M Seshu Bhavani[1],G Bhargavi[2],B Bhavani[3],K Lakshmi[4]

[1]Associate Professor;Department Of Computer Science And Engineering(AI&Ml) Bhoj Reddy Engineering College For Women Hyderabad India

[2,3,4]B.Tech Students; Department Of Computer Science And Engineering(AI&Ml) Bhoj Reddy Engineering College For Women Hyderabad India

Mail Id;bhargavigoud841@gmail.com[2],bommubhavani08@gmail.com[3],katravathlakshmi46@gmail.com[4]

ABSTRACT

Stock market trading requires timely decisions, accurate analysis, and continuous monitoring of dynamic financial data. Traditional trading methods often depend on manual interpretation of charts, market news, and historical trends, which can lead to delayed actions and emotionally influenced decisions. Many retail investors and beginner traders face difficulties in understanding market behavior, identifying profitable opportunities, and managing trading risks effectively.This paper presents an Agentic Trading Bot, an intelligent automated trading framework that combines Artificial Intelligence, Deep Learning, and explainable agent-based reasoning for stock market decision support. The proposed system employs advanced sequential learning models, including Long Short-Term Memory (LSTM) networks and Transformer-based architectures, to capture temporal dependencies and hidden patterns in historical stock price data for trend forecasting. The framework processes market information obtained from CSV datasets or live financial APIs and generates actionable signals such as Buy, Sell, or Hold based on predictive outcomes and predefined risk strategies.To improve transparency and user confidence, the system integrates an agentic reasoning layer using LangChain, which interprets model outputs and provides human-readable explanations for generated recommendations. A web-based interface developed with Streamlit enables users to upload datasets, monitor predictions, and interact with the trading assistant in real time.Experimental observations indicate that the proposed approach can reduce emotional bias, improve decision consistency, and enhance trading efficiency when compared with purely manual trading practices. The modular design of the system supports scalability, adaptability to multiple market conditions, and future integration with automated brokerage platforms. The Agentic Trading Bot offers a practical solution for intelligent financial decision-making for both novice and experienced traders.

Keywords: *Stock Market Prediction, Algorithmic Trading, Agentic AI, Deep Learning, LSTM, Transformer Model, LangChain, Explainable AI, Streamlit, Financial Forecasting.*

Introduction

Stock market trading is the activity of purchasing and selling financial instruments such as shares with the objective of earning returns from price fluctuations. It has become an essential component of the global financial system, attracting retail investors, institutional participants, and brokerage firms through digital trading platforms. Although the concept of buying at a lower price and selling at a higher price appears straightforward, achieving sustainable profitability requires continuous market observation, technical expertise, and timely decision-making.Modern financial markets generate enormous volumes of structured and unstructured data, including historical prices, intraday movements, trading volume, company announcements, macroeconomic indicators, and investor sentiment. Processing and interpreting this information manually is difficult, especially for individual traders. Human decisions are also frequently affected by psychological factors such as fear, greed, overconfidence, and hesitation, which may lead to inconsistent trading behavior and avoidable losses.To address these issues, this research proposes an Agentic Trading Bot, an intelligent decision-support system that automates market analysis and assists users in making rational trading choices. The framework uses Artificial Intelligence and Deep Learning techniques to study past and live market data, recognize hidden trends, and estimate possible future price movements. Based on these insights, the system provides practical recommendations in the form of Buy, Sell, or Hold signals.The proposed model incorporates advanced sequence-learning architectures such as Long Short-Term Memory (LSTM) networks and Transformer models, which are highly effective in understanding time-dependent financial data. These models can capture short-term volatility as well as long-range market dependencies, thereby improving forecasting capability. In addition, an agentic reasoning module can interpret model outcomes and explain the logic behind each recommendation, increasing transparency and user trust. **Existing System**

Current stock market trading systems mainly depend on manual decision-making supported by online brokerage platforms such as Zerodha, Groww, Robinhood, and similar services. These platforms provide users with trading access, charting tools, price alerts, and limited analytical indicators.

However, the final responsibility for selecting stocks, interpreting trends, and executing trades remains with the trader.In conventional practice, traders rely on chart patterns, candlestick behavior, moving averages, momentum indicators, and market news to anticipate future price direction. Some participants also use statistical approaches such as regression analysis, volatility models, and rule-based strategies. While these techniques may be useful in stable market conditions, they often struggle to capture nonlinear relationships, sudden sentiment shifts, and rapidly changing volatility patterns present in real-world markets.Automated trading systems have also gained popularity in recent years. Many of these systems execute trades based on predefined rules such as crossover signals, threshold triggers, or fixed technical conditions. More advanced platforms may use basic machine learning models for prediction. Despite offering automation, such systems frequently face several limitations. They may perform poorly when market behavior changes, require regular manual tuning, and often lack interpretability.Another major drawback of many existing solutions is the absence of explainable decision support. Users may receive a trading signal without understanding why it was generated, which reduces confidence and limits adoption among retail traders. Furthermore, several systems focus only on prediction accuracy while ignoring risk management, adaptability, and user interaction.

Literature Survey

Research in stock market forecasting and algorithmic trading has progressed rapidly with the adoption of Artificial Intelligence, Machine Learning, and Deep Learning methods. Earlier financial prediction systems were largely based on statistical models such as ARIMA, moving averages, and regression techniques. Although these methods provided useful baseline forecasts, their ability to model nonlinear relationships and rapidly changing market conditions was limited.Recent studies have shown that deep learning architectures offer stronger predictive performance for sequential financial data. Investigations on Long Short-Term Memory (LSTM) networks reported superior results over conventional time-series techniques because LSTM models can retain long-range dependencies and learn hidden temporal structures in stock price movements. These models are particularly effective when dealing with volatile and non-stationary market behavior. However, researchers also noted challenges such as overfitting, hyperparameter sensitivity, and the requirement for large training datasets.Further developments in recurrent neural networks demonstrated that sequential learning models can significantly improve short-term market trend prediction. Financial datasets contain delayed correlations, momentum effects, and periodic fluctuations that are difficult to capture using traditional approaches. Neural architectures with memory mechanisms have therefore become increasingly popular in automated trading research.Ensemble learning methods have also been widely studied for market direction forecasting. Random Forest, Gradient Boosting, and similar tree-based classifiers have shown stable performance when trained on technical indicators, price returns, and volume-related features. These approaches are valued for their robustness to noisy data, reduced risk of overfitting, and comparatively lower computational cost when compared with deep neural networks.Some researchers proposed hybrid systems that combine machine learning models with domain-specific knowledge such as seasonality, volatility cycles, and macroeconomic signals. These studies found that integrating feature engineering with predictive models can improve profitability and trade consistency. Hybrid methods are particularly useful in short-term speculative markets such as equities and foreign exchange trading.Reinforcement Learning (RL) introduced another major direction in algorithmic trading. Instead of predicting only future prices, RL systems learn optimal buy and sell policies by interacting with a simulated market environment and maximizing cumulative rewards. Such models can adapt dynamically to changing trends, transaction costs, and risk constraints. Despite their promise, RL-based trading agents often require careful reward design, extensive training time, and substantial computational resources.The concept of multi-agent systems has also gained attention in modern trading automation. In these frameworks, separate intelligent agents perform specialized tasks such as market forecasting, portfolio balancing, risk control, sentiment monitoring, and trade execution. Distributed agent structures improve modularity, scalability, and system resilience, though coordination among agents remains a technical challenge.Another important research direction is financial sentiment analysis. Studies using social media posts, news headlines, and investor discussions demonstrated that public sentiment can influence short-term stock movements. Natural Language Processing (NLP) techniques therefore provide a valuable supplementary signal when combined with numerical market indicators.

Methodology

The proposed Agentic Trading Bot is designed as an integrated intelligent framework that combines market data acquisition, predictive analytics, reasoning modules, and user interaction components. The methodology focuses on converting raw financial information into reliable trading recommendations through automated analysis.

Functional Requirements

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

The system includes a user management layer that supports secure registration, authentication, profile control, and API credential storage for brokerage or market data platforms. Users can upload CSV datasets, choose predefined stocks, and switch between simulation trading and live execution modes. The interface allows strategy selection, signal visualization, historical trade review, and performance analytics.A data ingestion module is responsible for collecting historical and real-time stock market data through APIs or uploaded files. It extracts useful variables such as open price, close price, high, low, trading volume, moving averages, RSI, MACD, and other technical indicators. The collected data is cleaned, normalized, and transformed into model-ready sequences.The decision-making engine applies deep learning models such as LSTM and Transformer networks to estimate future price trends and classify market states. Based on prediction confidence and risk thresholds, the system generates actionable recommendations in the form of Buy, Sell, or Hold signals. An agentic reasoning component interprets the model outputs and provides transparent explanations for each decision.The execution module records signals, updates simulated portfolios, or forwards approved trades to supported brokerage APIs. It also maintains logs for auditing, strategy evaluation, and future retraining.

Non-Functional Requirements

The platform must provide low-latency responses suitable for active trading environments. Market data retrieval, processing, and signal generation should occur within a few seconds to preserve decision relevance.Usability is a key requirement; therefore, the interface must remain intuitive for both novice and experienced users. Clear dashboards, interactive charts, and readable explanations are essential for adoption.Security measures include encrypted communication channels, secure storage of credentials, access control mechanisms, and protection against unauthorized usage.Scalability is required to support increasing numbers of users, multiple concurrent assets, and expanding data volumes. Cloud-ready deployment and modular services help maintain consistent performance.Reliability and availability are also critical. The system should include fault tolerance, scheduled backups, logging mechanisms, and recovery procedures to minimize downtime during trading hours.

Computational Requirements

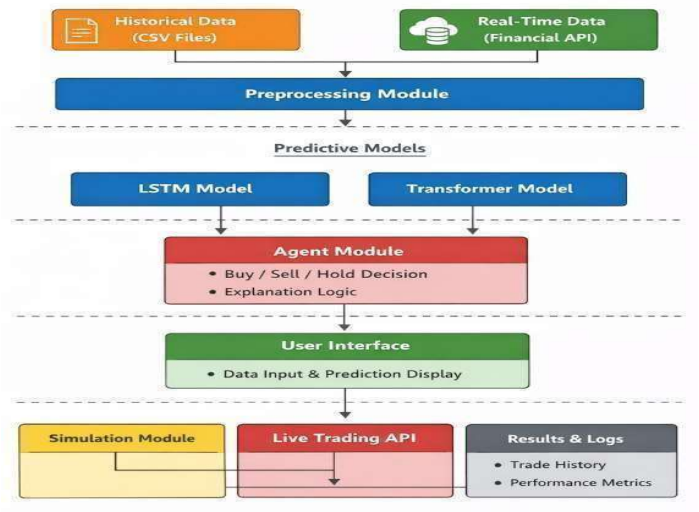
The implementation environment uses Python as the primary development language due to its extensive ecosystem for finance and AI applications. A Streamlit-based frontend enables rapid development of interactive dashboards. SQLite or similar lightweight databases can be used for storing users, transactions, logs, and model outputs.Core libraries include NumPy and Pandas for numerical operations and data handling, Scikit-learn for preprocessing and baseline models, TensorFlow or PyTorch for deep learning, LangChain for agentic workflows, and Matplotlib for visual analytics.Recommended hardware includes a modern multi-core processor, at least 8 GB RAM, SSD storage, and optional GPU acceleration for faster model training and hyperparameter tuning.

Design

Architecture

The proposed Agentic Trading Bot follows a modular and layered architecture to ensure maintainability, scalability, and efficient execution. The overall design separates user interaction, data processing, predictive intelligence, decision-making, and execution services into independent modules. This structure allows each component to be upgraded or optimized without affecting the complete system.

Software Architecture

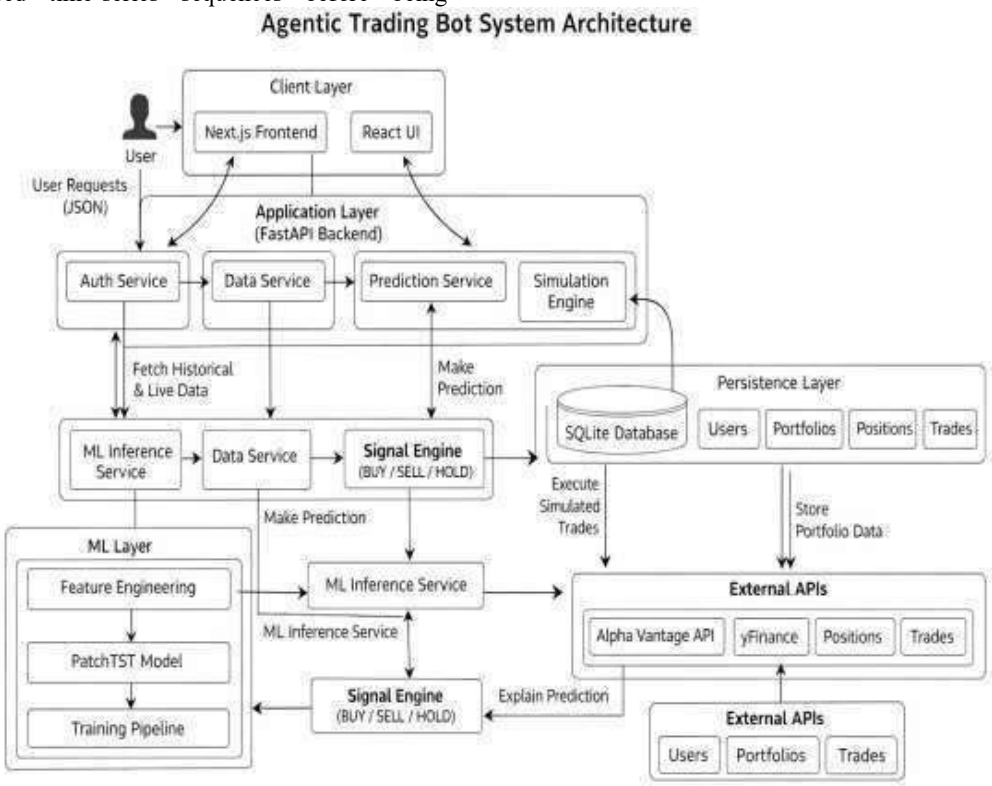


Fig;1 Software Architecture

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026

The software architecture consists of five major layers. The presentation layer is implemented using Streamlit and provides dashboards for user login, stock selection, file upload, signal monitoring, and report visualization. The application control layer manages workflows such as authentication, request routing, portfolio updates, and user session handling.The analytics layer handles data preprocessing, feature generation, technical indicator computation, and model execution. Historical and live data are transformed into structured time-series sequences before being passed to prediction models. The intelligence layer includes LSTM and Transformer networks for trend forecasting, along with an agentic reasoning engine that explains outputs and supports decision transparency.The persistence layer stores user profiles, logs, historical records, model metadata, and simulated trade history using SQLite or similar databases. An integration layer connects the system to external market APIs, brokerage services, and notification channels.

Technical Architecture



Fig;2 Technical Architecture

The technical architecture is built around Python-based technologies. The frontend interface is developed with Streamlit, while backend services run through Python modules and optional Flask-based APIs. Real-time market data is collected through providers such as Yahoo Finance or Alpha Vantage. Machine learning and deep learning pipelines are implemented using Scikit-learn, TensorFlow, or PyTorch.The reasoning framework uses LangChain to create explainable workflows that interpret model predictions and generate natural language insights. Visualization modules use Matplotlib and related libraries for chart rendering. Deployment can be performed on local systems or cloud infrastructure depending on scale requirements.

Implementation

Technologies

The Agentic Trading Bot is implemented using a Python-centered ecosystem because of its strong support for data science, machine learning, and financial analytics. The development environment includes Python libraries for data processing, model training, visualization, and web deployment.TensorFlow and PyTorch are used to build deep learning architectures such as LSTM and Transformer networks. LangChain provides an agentic reasoning layer that converts raw predictions into understandable explanations. Streamlit is used to create an interactive web interface where users can upload datasets, monitor live predictions, and review portfolio performance. Optional Flask APIs can be used for modular backend deployment.Market data is obtained from CSV files or real-time providers such as Yahoo Finance and Alpha Vantage. Pandas and NumPy manage tabular data operations, while Scikit-learn supports

preprocessing, scaling, and baseline learning algorithms.

Preprocessing Steps

The preprocessing pipeline begins with data collection from uploaded files or external APIs. Typical attributes include open, high, low, close prices, adjusted close values, and trading volume. Real-time feeds may also be integrated for active prediction mode.Data cleaning removes duplicate records, handles missing values using interpolation or forward filling, and filters inconsistent observations. Once cleaned, date fields are standardized and records are arranged chronologically.Feature engineering is then applied to improve predictive capability. Common indicators include Moving Average, Exponential Moving Average, Relative Strength Index, MACD, Bollinger Bands, and momentum-based variables. These engineered features provide additional information beyond raw prices.Normalization and scaling are performed using MinMaxScaler or StandardScaler so that features fall within suitable numeric ranges. This improves gradient stability during neural network training.The processed dataset is converted into sliding-window sequences. For example, the previous ten trading sessions may be used to predict the next day's movement. Finally, the data is divided into training and testing subsets for model validation and generalization analysis.

Algorithmic Workflow

The application starts by loading a pretrained model and initializing user session variables. Through the dashboard, the user chooses either CSV upload mode or live market mode. Input data is collected and passed through preprocessing functions.After preprocessing, the model generates trend forecasts or directional probabilities. The decision engine interprets these outputs and assigns a Buy, Sell, or Hold recommendation according to confidence thresholds and risk rules.The reasoning module then produces a human-readable explanation describing why the signal was generated. Results are shown through charts, tables, and performance summaries. Users may continue in simulation mode or connect supported APIs for live execution.

Testing

Testing is an essential stage in validating the reliability, correctness, and efficiency of the Agentic Trading Bot. Since the system handles financial decisions, careful evaluation is necessary to ensure accurate outputs, secure operation, and stable performance under varying workloads.The testing process covers all major modules, including data ingestion, preprocessing, forecasting, signal generation, visualization, simulation, and trade execution. Both isolated module testing and end-to-end testing are conducted.

Test Objectives

The primary objective of testing is to confirm that every module performs according to design specifications. The system must correctly process uploaded or live data, generate valid Buy, Sell, or Hold signals, and display understandable outputs.Additional goals include validating model accuracy, verifying seamless interaction among components, ensuring low response time, protecting sensitive credentials, and confirming usability for non-technical users.

Stages of Testing

Unit testing is performed on individual components such as preprocessing functions, feature generators, API connectors, and prediction methods. This helps detect coding errors early in development.Integration testing checks whether modules exchange data correctly and function together without workflow failures. It verifies compatibility among ingestion, analytics, prediction, and execution services.System testing evaluates the complete application in a realistic environment. It confirms that user interactions, signal generation, dashboards, and simulations operate as expected.Performance testing measures execution speed, responsiveness, and system stability under heavy datasets or concurrent requests. The target response time for real-time decisions remains within a few seconds.

Types of Testing

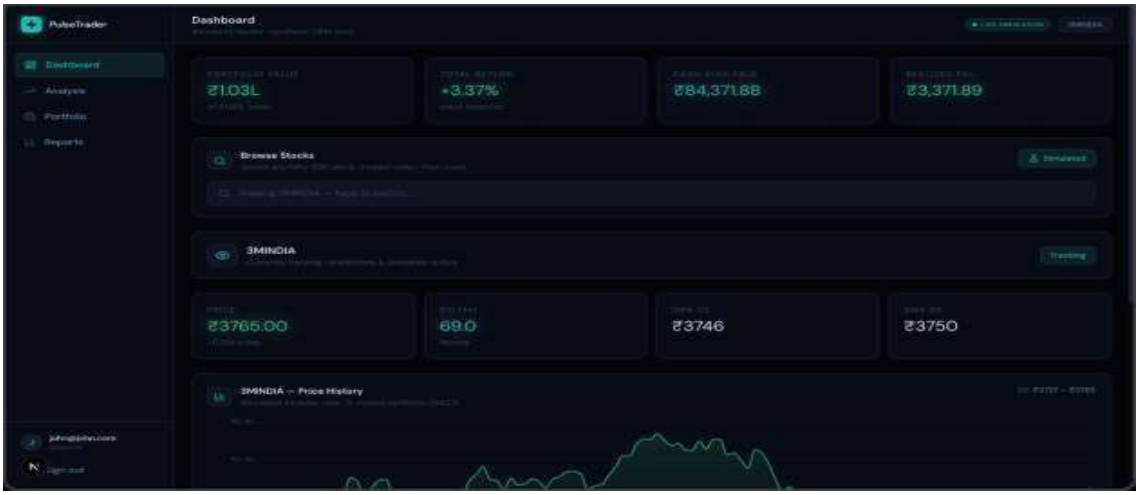
White box testing focuses on internal program logic, conditional branches, loops, and algorithm execution. It is useful for validating preprocessing pipelines and model workflows.Black box testing examines the system externally without reviewing internal code. Inputs such as CSV uploads, stock symbols, and user actions are supplied, and outputs are compared against expected behavior.Security testing verifies encryption, safe credential handling, and controlled access. Usability testing ensures that navigation and dashboard interaction remain simple and intuitive.

Test Case Summary

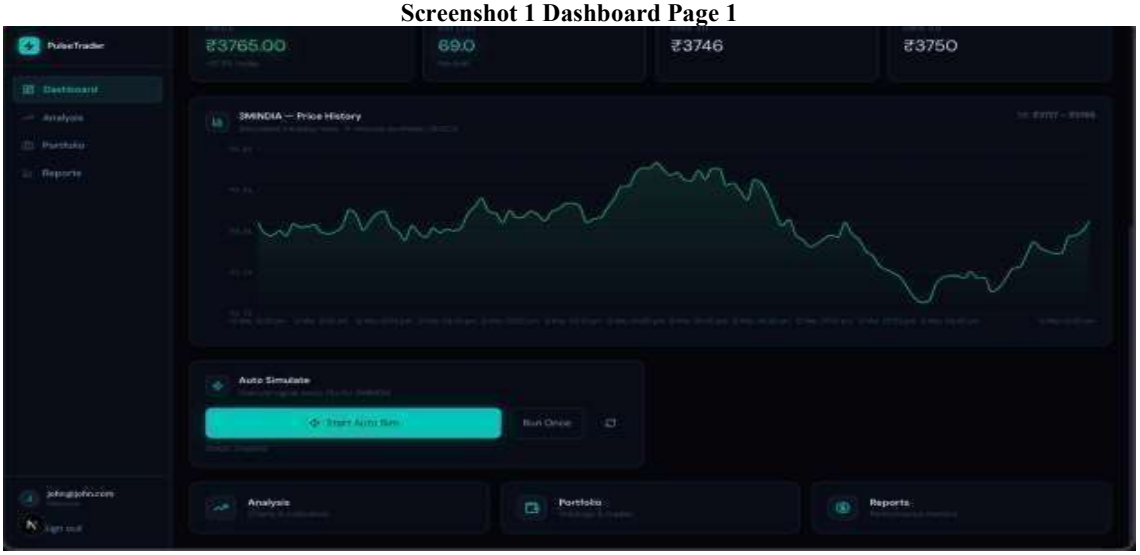
Multiple scenarios were tested successfully, including valid CSV uploads, invalid file rejection, real-time data retrieval, preprocessing accuracy, prediction generation, explanation rendering, graph visualization, simulation mode, live trade triggering, warning messages for missing input, and stable performance under larger datasets.Security checks confirmed safe handling of API keys, while stress testing showed stable operation during prolonged usage. The successful completion of these tests indicates that the proposed Agentic Trading Bot is suitable for practical deployment and further research enhancement.

RESULTS

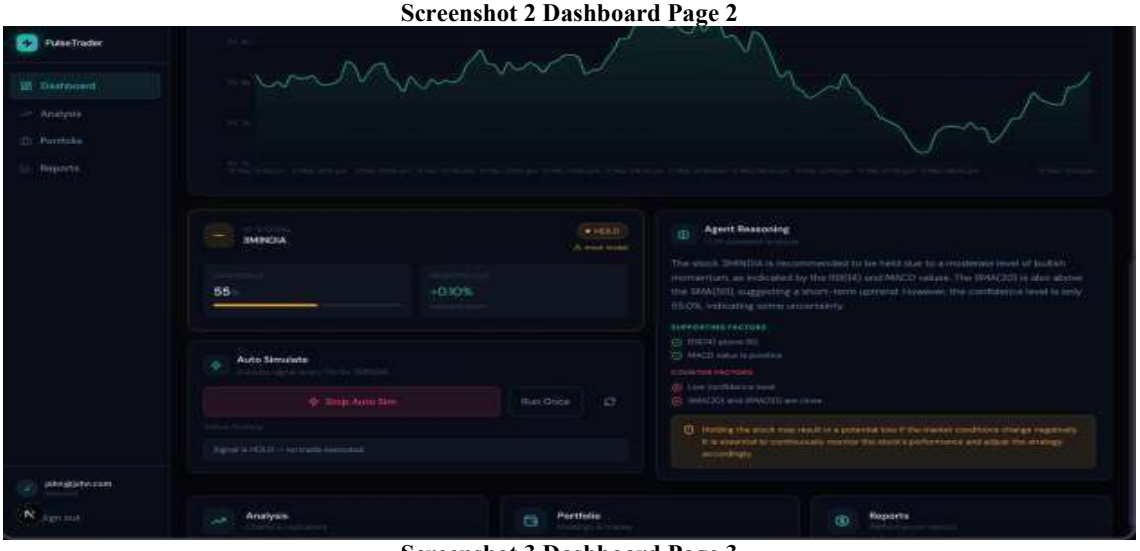
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



Screenshot 1 Dashboard Page 1

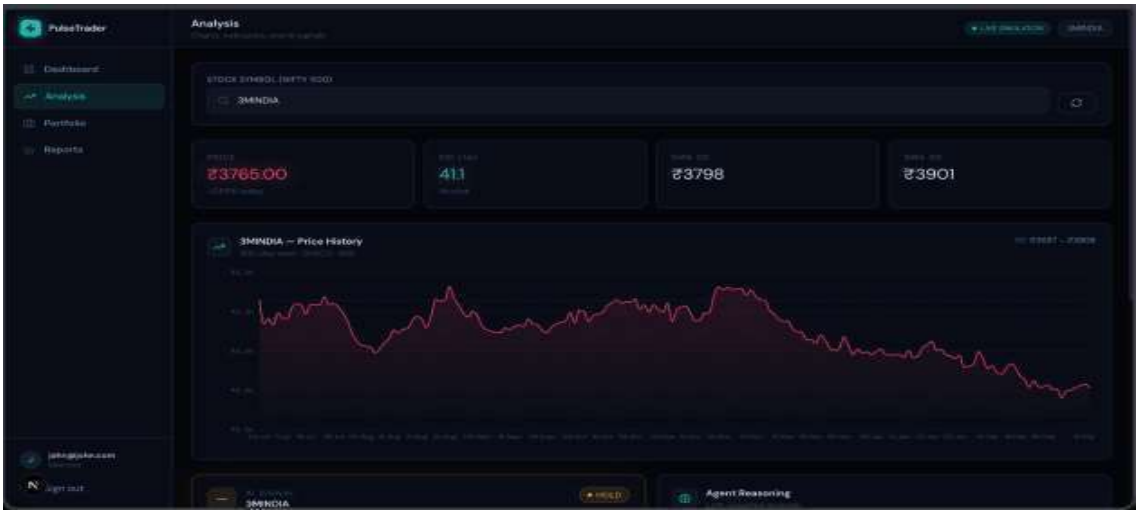


Screenshot 2 Dashboard Page 2

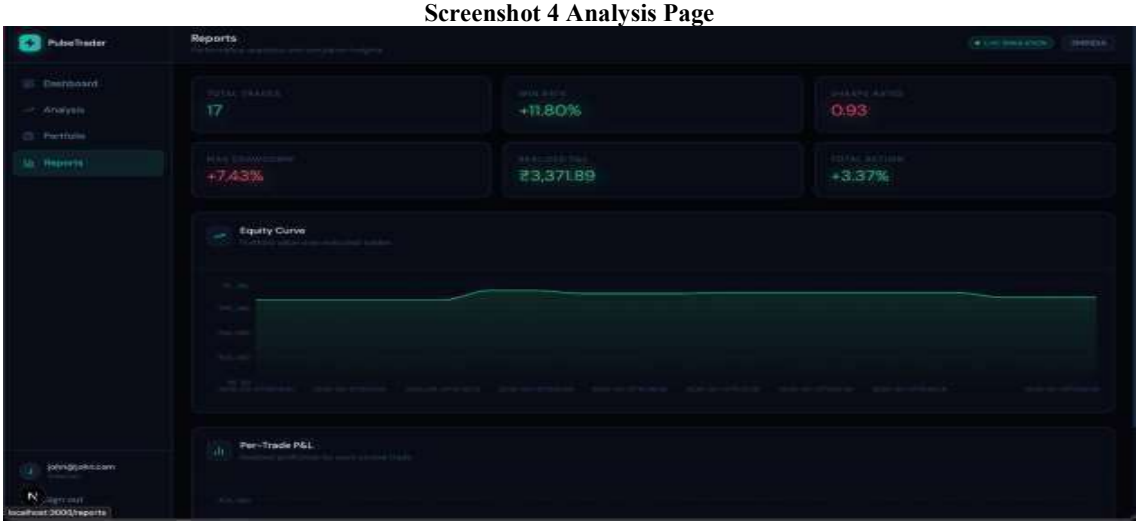


Screenshot 3 Dashboard Page 3

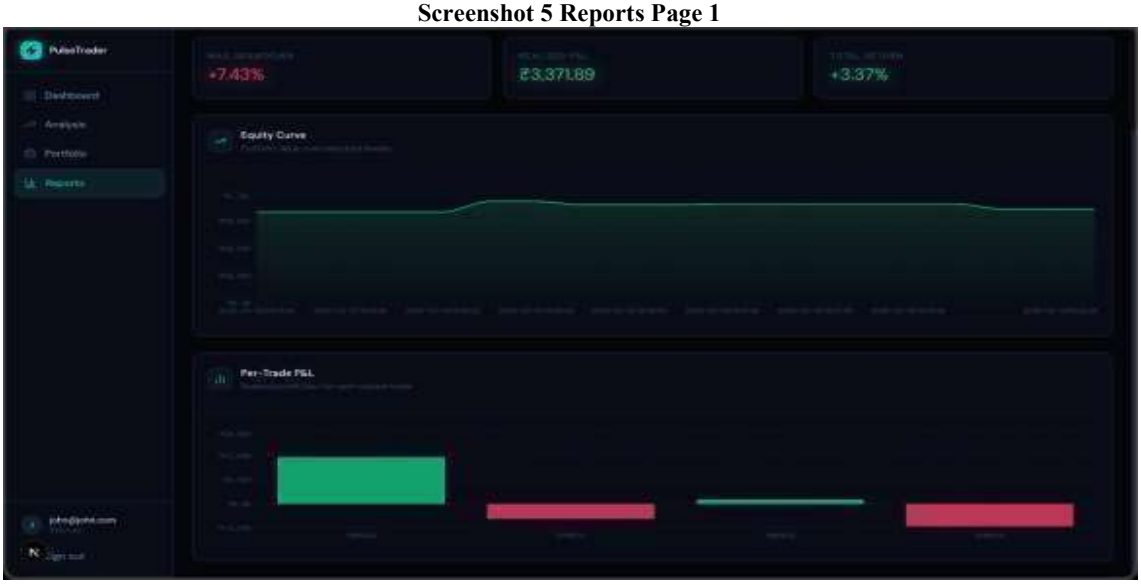
International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 14, Issue 2, 2026



Screenshot 4 Analysis Page



Screenshot 5 Reports Page 1



Screenshot 6 Reports Page 2

Conclusion

The proposed Agentic Trading Bot demonstrates how modern Artificial Intelligence and Deep Learning methods can be effectively applied to stock market decision support. The system successfully combines data acquisition, preprocessing, predictive

analytics, explainable reasoning, and trading signal generation within a unified intelligent platform. By integrating sequence-learning models such as LSTM and Transformer networks, the framework is capable of identifying temporal market patterns and estimating future price movement with improved consistency.The inclusion of both simulation and live trading modes increases the practical value of the system. Simulation mode enables users to evaluate strategies and understand portfolio behavior without financial exposure, while live mode supports real-time decision assistance in dynamic market environments. This dual-mode structure makes the platform useful for beginners seeking learning opportunities as well as experienced traders requiring analytical support.Another important contribution of the project is the reduction of emotional bias in trading decisions. Human behavior often leads to impulsive actions caused by fear, greed, or hesitation. The automated reasoning process helps replace subjective judgment with data-driven recommendations, thereby improving discipline and operational efficiency. In addition, the explainable agentic layer enhances user trust by clearly presenting the basis for each Buy, Sell, or Hold signal.Experimental evaluation and system testing indicate that the proposed framework is scalable, reliable, and suitable for real-world deployment with further refinement. Overall, the study confirms that AI-enabled trading assistants can significantly improve the speed, transparency, and effectiveness of traditional trading approaches, making them highly relevant to the evolving digital finance ecosystem.

Future Scope

Future development of the Agentic Trading Bot can focus on several advanced enhancements. Reinforcement Learning techniques may be integrated to allow the system to learn adaptive trading strategies directly from market interactions and optimize long-term returns under changing conditions.Prediction quality can be further improved by incorporating sentiment intelligence from financial news, analyst reports, and social media discussions. Combining textual sentiment with price-based indicators may provide earlier signals of market movement.A multi-agent framework can also be developed in which independent intelligent agents manage forecasting, portfolio balancing, trade execution, and risk control collaboratively. Such distributed intelligence would improve modularity, robustness, and scalability.The platform can be extended beyond equity markets to include foreign exchange, cryptocurrencies, commodities, and derivatives. This would increase its usefulness across diverse financial sectors.Additional research may emphasize advanced risk management features such as dynamic stop-loss policies, volatility-aware position sizing, diversification engines, and automated hedging mechanisms. Cloud deployment, mobile access, and broker integration are also promising directions for commercial implementation.

References

[1] T. Fischer and C. Krauss, “Deep learning with long short-term memory networks for financial market predictions,” *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[2] K. Kim, “Financial time series forecasting using LSTM neural networks,” *IEEE Access*, vol. 7, pp. 123–130, 2019.

[3] S. Basak, S. Kar, S. Saha, L. Khaidem, and S. R. Dey, “Predicting the direction of stock market prices using tree-based classifiers,” *The North American Journal of Economics and Finance*, vol. 47, pp. 552–567, 2019.

[4] A. Booth, E. Gerding, and F. McGroarty, “Automated trading with performance-weighted random forests and seasonality,” *Expert Systems with Applications*, vol. 41, no. 8, pp. 3651–3661, 2014.

[5] J. Moody and M. Saffell, “Learning to trade via direct reinforcement,” *IEEE Transactions on Neural Networks*, vol. 12, no. 4, pp. 875–889, 2001.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., MIT Press, 2018.

[7] M. Lopez de Prado, *Advances in Financial Machine Learning*, Wiley, 2018.

[8] J. Bollen, H. Mao, and X. Zeng, “Twitter mood predicts the stock market,” *Journal of Computational Science*, vol. 2, no. 1, pp. 1–8, 2011.

[9] F. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.

[10] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.