© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718363

Automated Algo Trading System with Subscription-Based Paper Trading

ABHA DHIRENDRA SINGH[1], DR. PRAKASH KENE[2]
[1]*MCA, PES's Modern College of Engineering Pune, India*
[2]*Associate Professor, MCA Department PES's Modern College of Engineering Pune, India*

Abstract- ***The rapid evolution of financial technology (FinTech) has transformed stock market participation for retail investors across the globe. Despite the proliferation of digital trading platforms, beginners continue to face significant financial risks due to insufficient practical knowledge and the absence of structured simulation environments. This paper presents the design and implementation of an Automated Algo Trading System with Subscription-Based Paper Trading—a web-based platform enabling users to simulate stock market trades using virtual funds in a completely risk-free environment. The system integrates a subscription-based access model (Free and Premium tiers) to deliver a scalable, monetizable, and beginner-friendly trading education platform. Built using Python (FastAPI backend), React.js (frontend), and Firebase (authentication and real-time database), the system supports real-time market data display, virtual Buy/Sell operations, portfolio management, profit/loss analytics, and structured reporting. Evaluation across multiple functional test cases demonstrates the system's effectiveness in bridging the gap between theoretical financial knowledge and practical trading experience, particularly for novice retail investors in emerging markets such as India.***

Keywords— ***Algorithmic Trading, Paper Trading, FinTech, Subscription Model, Firebase, React.js, Portfolio Management, Stock Market Simulation, FastAPI, Machine Learning***

I. INTRODUCTION

The global financial technology sector has witnessed unprecedented growth over the last decade, fundamentally democratizing access to stock markets through mobile and web-based trading platforms [1]. In India alone, the number of registered retail investors surpassed 140 million by 2024, driven primarily by platforms such as Zerodha, Upstox, and Grow [2]. Despite this remarkable democratization, a large proportion of novice traders continue to incur significant financial losses during their initial years of participation due to limited market understanding, emotional decision-making, and absence of hands-on practice environments.

Traditional trading education relies predominantly on classroom-based theoretical instruction and passive content consumption, which fails to prepare users for the inherent complexity and volatility of live financial markets. The high-stakes nature of real-money trading creates a steep learning curve that discourages many prospective investors before they develop meaningful competence. While some platforms offer paper trading—simulated trading using virtual funds—these features are often poorly integrated, feature-limited, and inaccessible to free-tier users [3].

The absence of structured, gamified, and progressively unlockable simulation environments represents a critical gap in the retail investor education ecosystem. Furthermore, existing FinTech platforms rarely combine simulation capabilities with sustainable monetization architectures, making it difficult to maintain platform quality while serving a broad user base.

To address these challenges comprehensively, this paper proposes an Automated Algo Trading System that integrates: (i) a paper trading engine enabling risk-free simulation with real-time market data, (ii) a subscription-based access model that progressively gates advanced features, (iii) a modern responsive web interface built with React.js for seamless cross-device experience, (iv) a Python FastAPI backend implementing the core trading logic, (v) Firebase for scalable authentication and real-time data persistence, and (vi) structured reporting and analytics for user performance evaluation.

The system was conceptualized and developed during an industrial internship at Landmark Techedge Pvt. Ltd., Pune, providing a real-world deployment context. The remainder of this paper is organized as follows: Section II reviews related literature; Section III describes limitations of existing systems; Section IV presents the proposed system architecture; Section V details the system design; Section VI discusses implementation; Section VII presents testing results; Section VIII outlines future enhancements; and Section IX concludes the paper.

II. LITERATURE REVIEW

A. Paper Trading and Financial Simulation

Research by Bauer et al. [4] demonstrated that simulation-based financial learning significantly improves investment decision-making quality among novice investors compared to traditional instruction. Empirical studies across business school cohorts consistently show that students who engage with virtual trading environments develop superior risk assessment skills and portfolio construction strategies within shorter time frames.

Chandra and Mehta [1] further established that interactive simulation platforms create measurable improvements in financial literacy metrics, including understanding of market microstructure, order types, and price formation mechanisms. The cognitive engagement facilitated by real-time feedback loops in simulation environments has been linked to significantly higher knowledge retention rates compared to passive learning modalities.

B. Algorithmic Trading Systems

Algorithmic trading refers to the systematic use of computer programs to execute trades based on predefined rules, quantitative models, and statistical signals [5]. Chan's foundational work on algorithmic trading strategies outlines the mathematical underpinnings of mean reversion, momentum, and arbitrage strategies that form the basis of most modern quantitative trading systems.

While institutional investors have leveraged algorithmic trading for decades, retail-level access has grown substantially with the availability of broker APIs from platforms such as Zerodha Kite Connect, Upstox Pro, and Angel Broking SmartAPI. However, the tooling required to design, backtest, and simulate algorithmic strategies remains primarily oriented toward technically sophisticated users, creating a significant barrier for retail investors with limited programming backgrounds.

C. Subscription-Based SaaS in FinTech

The Software-as-a-Service (SaaS) model has become the dominant commercial paradigm in contemporary FinTech applications [6]. Subscription-based access architectures enable platforms to offer meaningful free-tier entry points while monetizing advanced capabilities through premium tiers. Reshef and Galor's analysis of FinTech SaaS models demonstrates that tiered subscription structures achieve higher user lifetime values while maintaining lower customer acquisition costs compared to transactional monetization models.

Prominent examples of successful tiered FinTech SaaS implementations include TradingView, which segments charting capabilities across free, Pro, Pro+, and Premium tiers, and Streak, which gates algorithmic strategy automation behind subscription plans. These models demonstrate the commercial viability of the proposed architecture.

D. Machine Learning in Trading

Recent advances in deep learning have opened new avenues for predictive modeling in financial markets. Long Short-Term Memory (LSTM) neural networks, first applied to financial time series by Hochreiter and Schmidhuber [9], have demonstrated competitive performance in short-term price direction forecasting. Random Forest ensemble models offer complementary strengths in handling non-linear feature interactions and providing feature importance rankings for technical indicators.

The integration of machine learning components into trading platforms, however, requires careful consideration of overfitting risks, data snooping bias, and transaction cost modeling to ensure that backtested performance translates meaningfully to live or simulated trading environments.

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718363

E. Research Gap

Despite extensive literature on paper trading, algorithmic systems, and SaaS monetization independently, a clear gap exists in platforms that simultaneously: (1) combine structured paper trading with subscription-based monetization for emerging market retail investors, (2) provide integrated learning pathways alongside trading simulation to guide novice users, (3) implement a full-stack web architecture suitable for rapid iteration and cloud deployment, and (4) offer a foundation extensible to AI-powered predictive analytics. This work addresses all four dimensions in a unified system.

III. EXISTING SYSTEM AND LIMITATIONS

Current trading platforms such as Zerodha, Upstox, and Angel Broking provide powerful tools for experienced traders but fall significantly short in serving the educational needs of beginner retail investors. A comparative analysis of leading platforms reveals systemic gaps across multiple dimensions critical for novice user success.

Zerodha Kite, despite being India's leading retail broker by active client count, restricts its paper trading simulation to users who have specifically enabled the feature through developer account settings—a process inaccessible to non-technical users. Upstox similarly provides API-based algorithmic trading capabilities that require programming knowledge, effectively excluding the majority of its user base from meaningful practice.

TABLE I. COMPARATIVE ANALYSIS OF EXISTING TRADING PLATFORMS

Platform Feature	Zerodha	Upstox	Groww	Proposed System
Paper Trading	Limited	API Only	None	Full Support
Learning Pathways	Varsity (External)	None	None	Integrated
Subscription Tiers	None	API Plans	None	Free & Premium
Beginner UI	Moderate	Complex	Simple	Guided
Real-time P&L	Yes	Yes	Limited	Yes
AI Predictions	None	None	None	Planned
Report Generation	Basic	API Only	Basic	Structured

The absence of structured guidance within simulation environments is particularly consequential for novice investors. Without contextual explanations of market behavior, order execution mechanics, and risk management principles integrated directly into the trading interface, users are left to develop market intuition through trial and error alone—an inefficient and potentially discouraging learning process.

IV. PROPOSED SYSTEM

A. System Overview

The proposed system is a web-based Algorithmic Trading Platform with Subscription-Based Paper Trading, structured around four core pillars: Education, Simulation, Analytics, and Monetization. The platform is designed as a progressive learning environment where users advance from market observation through paper trading practice to strategy development and performance analysis.

The Education pillar delivers contextual market knowledge through integrated explanatory content, real-time data visualization, and guided onboarding flows. The Simulation pillar provides a fully functional paper trading engine with virtual capital allocation, order execution, and portfolio tracking.

The Analytics pillar offers comprehensive reporting on trading performance, including P&L attribution, trade history analysis, and portfolio composition metrics. The Monetization pillar implements a sustainable SaaS revenue model through tiered subscription access control.

B. Objectives

The primary objectives of the proposed system are as follows: (1) provide a risk-free paper trading environment that accurately simulates real market conditions using live or near-real-time price feeds; (2) enable users to test and evaluate algorithmic trading strategies without financial exposure; (3) implement a tiered subscription model distinguishing Free and Premium access levels to ensure platform sustainability; (4) deliver structured performance analytics supporting evidence-based improvement of trading strategies; (5) ensure architectural scalability to support concurrent users through cloud-native infrastructure; and (6) establish a modular codebase extensible to planned AI and real-time data enhancements.

C. System Architecture

The system follows a three-tier client-server architecture, separating presentation logic, business logic, and data persistence into independent, loosely coupled layers. This architectural pattern promotes maintainability, scalability, and independent deployment of each tier.

The presentation tier is implemented as a React.js single-page application (SPA) communicating with the backend exclusively through RESTful API calls. The business logic tier is implemented in Python using the FastAPI framework, which provides high-performance asynchronous request handling with automatic OpenAPI documentation generation. The data persistence tier combines Firebase Firestore for NoSQL document storage and Firebase Authentication for identity management.

This decoupled architecture enables horizontal scaling of the backend tier independently of the frontend, allowing the platform to handle increasing concurrent user loads through container orchestration without frontend redeployment.

V. SYSTEM DESIGN

A. Data Model and Entity Relationships

The data model of the proposed system is organized around five core entities: User, Subscription, Trade, Portfolio, and Stock. The User entity stores registration information, authentication state, and a foreign key reference to the active Subscription record. The Subscription entity defines the plan type (Free or Premium), activation timestamp, expiry date, and the set of feature flags unlocked by the plan.

The Trade entity records all virtual Buy and Sell operations, capturing the stock symbol, quantity, execution price, trade direction, timestamp, and a foreign key to the executing User. The Portfolio entity maintains a real-time aggregation of current holdings, average cost basis per holding, total virtual capital deployed, and available virtual balance. The Stock entity caches market data retrieved from external REST APIs, storing the symbol, last trade price, daily high/low, and volume data.

B. Use Case Analysis

The system supports two primary actor roles: the Authenticated User and the Administrator. The Authenticated User use cases encompass: Register, Login/Logout, View Real-Time Market Data, Execute Paper Trade (Buy/Sell), View Portfolio Summary, Analyze P&L Performance, Generate Reports, and Upgrade Subscription. The Administrator uses cases include: Manage User Accounts, Monitor Platform Activity, Configure Subscription Plans, and Review System Health Metrics.

The subscription access control mechanism is enforced at the API gateway level, with each protected endpoint validating the requesting user's subscription tier before processing. This server-side enforcement prevents circumvention of access controls through client-side manipulation.

C. Trading Engine Design

The paper trading engine implements a simplified order matching model suitable for simulation purposes. Market orders are executed immediately at the last known price of the selected stock, simulating the execution behavior of a live market order under

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718363

normal liquidity conditions. Limit orders are queued in a virtual order book and matched when the market price crosses the specified limit threshold.

Upon order execution, the engine performs atomic updates to three Firestore documents: the Trade record (new document creation), the Portfolio holdings (quantity and cost basis update), and the User's virtual balance (debit for Buy orders, credit for Sell orders). Firestore's transaction API ensures these updates are applied atomically, preventing race conditions in concurrent trading scenarios.

D. Subscription Access Control

The subscription architecture implements a capability-based access control model in which each subscription tier is associated with a predefined set of feature capabilities. Free tier capabilities include market data viewing, basic portfolio summary, and limited trade history access. Premium tier capabilities extend the Free tier with unlimited paper trading, advanced technical indicator overlays, full report generation, and early access to AI prediction features.

Capability enforcement is implemented as a middleware layer in the FastAPI backend that decodes the user's Firebase authentication token, retrieves the associated subscription record from Firestore, and validates the required capability before delegating to the route handler. This centralized enforcement ensures consistent access control across all API endpoints.

VI. IMPLEMENTATION

A. Technology Stack

TABLE II. TECHNOLOGY STACK SUMMARY

Layer	Technology	Purpose
Frontend	React.js 18	Dynamic, responsive SPA
State Mgmt	Redux Toolkit	Global application state
Backend	Python FastAPI	Business logic & trading engine
Database	Firebase Firestore	NoSQL real-time data storage
Auth	Firebase Auth	Secure identity management
Market Data	REST APIs (NSE)	Live/simulated stock feeds
Charts	Recharts / D3.js	Candlestick & analytics charts
Hosting	Firebase Hosting	CDN-backed SPA deployment
Dev Tools	VS Code, Postman	Development & API testing

B. Frontend Implementation

The React.js frontend is structured as a component-based SPA with client-side routing implemented through React Router v6. The application state is managed through Redux Toolkit, with separate slices for authentication state, market data cache, portfolio data, and subscription status. Component styling employs a combination of CSS Modules for component-scoped styles and a shared design token system ensuring visual consistency across screens.

Real-time price updates are delivered through Server-Sent Events (SSE) from the FastAPI backend, which subscribes to market data feeds and pushes price updates to connected clients at configurable intervals. Candlestick charts are rendered using Recharts with custom tooltip components displaying OHLCV (Open, High, Low, Close, Volume) data for each time period.

C. Backend Implementation

The Python FastAPI backend implements the RESTful API with automatic request validation through Pydantic data models. API endpoints are organized into four routers: authentication, market data, trading, and subscription management. The trading router implements the paper trading engine with comprehensive input validation, virtual balance management, and atomic Firestore transactions.

The market data service implements an intelligent caching layer that stores retrieved stock prices in Firestore with configurable TTL (Time-to-Live) values, reducing external API call frequency while maintaining price freshness within acceptable

© MAY 2026 | IRE Journals | Volume 9 Issue 11 | ISSN: 2456-8880
DOI: https://doi.org/10.64388/IREV9I11-1718363

tolerances for simulation purposes. Cache invalidation is triggered by TTL expiry or explicit refresh requests from authenticated users.

D. Key Modules

1) Authentication Module: Implemented using Firebase Authentication with email/password-based registration and Google OAuth as an alternative login provider. User sessions are managed through Firebase ID tokens, which are validated on every authenticated API request through the FastAPI authentication middleware.

2) Subscription Module: Users are assigned a Free plan on registration. The Premium upgrade flow integrates with payment gateway processing (Razor pay in the planned enhancement) and updates the user's subscription record in Firestore upon successful payment confirmation. Subscription tier controls are applied throughout the platform via the capability middleware.

3) Paper Trading Engine: The core module of the system. Users select a stock from the market scanner, specify order direction (Buy/Sell), enter quantity, and optionally set a limit price. The engine validates sufficient virtual balance for Buy orders, checks existing holdings for Sell orders, executes the trade, and returns a comprehensive execution confirmation including updated balance and portfolio metrics.

4) Market Data Service: Fetches live or simulated stock data via REST APIs from NSE India's public endpoints and supplementary data providers. Displays candlestick charts for technical trend analysis. A Manual Scanner component allows targeted searches by stock symbol with autocomplete powered by a local symbol index.

5) Portfolio Management: Tracks current holdings across all open positions, maintains available virtual balance, and dynamically calculates mark-to-market unrealized P&L based on current market prices. Realized P&L is computed cumulatively from the trade history for closed positions.

6) Report Generation Module: Generates four structured report types exportable as formatted documents: Trade History Report, Profit/Loss Report, Portfolio Report, and Subscription Report. Reports support date range filtering and are rendered in both screen-optimized and print-optimized formats.

VII. RESULTS AND EVALUATION

A. System Testing

The system was subjected to comprehensive functional testing across all primary user workflows. Test cases were designed to validate both positive (happy path) and negative (error path) scenarios for each module, with particular attention to boundary conditions in the trading engine's balance validation logic and the subscription access control middleware.

TABLE III. FUNCTIONAL TESTING RESULTS

Module	Test Case	Input Condition	Result
Authentication	Valid login	Correct credentials	Pass
Authentication	Invalid login	Wrong password	Pass
Subscription	Free-tier gate	Premium feature, Free user	Pass
Subscription	Premium access	Premium feature, Premium user	Pass
Trading Engine	Buy execution	Sufficient balance	Pass
Trading Engine	Insufficient funds	Balance < order cost	Pass
Trading Engine	Sell execution	Holdings > 0	Pass
Portfolio	Real-time P&L	Price change event	Pass
Reports	Trade history	Date range filter	Pass
Market Data	Live price display	REST API active	Pass
Market Data	API failure fallback	Simulated API down	Pass

B. Performance Metrics

Performance evaluation of the system focused on three key metrics: API response latency, concurrent user capacity, and data consistency under concurrent trading operations. API response latency was measured across 1,000 sequential requests for each endpoint category, yielding mean latencies of 45ms for market data queries, 78ms for trade execution requests, and 32ms for portfolio summary retrieval.

Concurrent user capacity testing was conducted using Apache JMeter with simulated load profiles of 50, 100, and 200 simultaneous users. The system maintained sub-200ms response times for 95th percentile latency at 100 concurrent users, with graceful degradation to sub-500ms at 200 concurrent users before requiring horizontal scaling.

TABLE IV. PERFORMANCE EVALUATION RESULTS

Metric	50 Users	100 Users	200 Users
Avg. Response (ms)	42	87	213
95th Pct (ms)	78	165	487
Error Rate (%)	0.0	0.2	1.1
Throughput (req/s)	118	95	72

C. User Interface Highlights

The user interface was designed following material design principles with a dark-themed trading dashboard inspired by professional trading terminals. Key interface components include: a real-time market ticker strip displaying live prices for watchlist stocks, a full-featured candlestick charting panel with configurable timeframes, a one-click paper trade execution panel with order confirmation, a portfolio summary card displaying virtual balance and aggregate P&L, and a subscription status indicator with upgrade prompt for Free tier users.

VIII. PROPOSED ENHANCEMENTS

A. Real-Time Market Data Integration

The current implementation uses cached market data refreshed at configurable intervals. A planned enhancement involves direct WebSocket integration with NSE India's official market data feed, enabling tick-by-tick price updates with sub-second latency. This enhancement will require a dedicated market data microservice with fan-out capability to broadcast price updates to all connected clients simultaneously.

B. AI-Powered Predictive Analytics

The planned AI enhancement module will integrate LSTM (Long Short-Term Memory) neural networks for short-term price direction forecasting and Random Forest models for technical indicator-based signal generation. Models will be trained on historical NSE price data with feature engineering incorporating RSI, MACD, Bollinger Bands, and volume-weighted moving averages as input features.

Model outputs will be presented as probability-weighted directional signals rather than deterministic price predictions, explicitly communicating prediction uncertainty to users and emphasizing their educational rather than advisory nature. All AI-generated signals will carry prominent disclaimers consistent with SEBI's algorithmic trading guidelines.

C. Backtesting Engine

A backtesting module will enable users to evaluate the historical performance of custom trading strategies on archived market data without real or virtual capital exposure. The backtesting engine will implement realistic transaction cost modeling, including brokerage fees, STT (Securities Transaction Tax), and exchange charges applicable to Indian equity markets, ensuring that backtested results account for the frictional costs that significantly impact net trading returns.

D. Mobile Application

A cross-platform mobile application will be developed using React Native, sharing business logic components with the existing React.js web frontend through a shared component library. The mobile application will introduce push notification support for price alerts, trade confirmations, and subscription renewal reminders, improving platform engagement and user retention metrics.

E. Gamification and Social Features

A competitive leaderboard system will rank users by simulated portfolio returns over configurable time

periods, introducing gamification mechanics that increase platform engagement and extend average user session durations. Social features including public portfolio sharing and copy trading simulation will enable collaborative learning through peer performance benchmarking.

IX. CONCLUSION

This paper presented the design, implementation, and evaluation of an Automated Algo Trading System with Subscription-Based Paper Trading—a comprehensive web-based platform addressing critical gaps in retail investor education and simulation infrastructure. The proposed system successfully bridges the divide between theoretical financial knowledge and practical trading experience by delivering a structured, risk-free simulation environment specifically designed for novice retail investors.

The integration of a tiered subscription-based access model ensures the platform's commercial sustainability while maintaining meaningful free-tier accessibility for beginner users. The three-tier architecture combining React.js, Python FastAPI, and Firebase provides a scalable, maintainable, and cloud-deployable foundation that demonstrated sub-200ms response latency at 100 concurrent users in performance evaluation.

Comprehensive functional testing across all system modules confirmed correct implementation of the paper trading engine, subscription access control, portfolio management, and report generation capabilities. The modular architecture supports clean separation of concerns between the trading simulation, market data, authentication, and subscription management subsystems.

With planned enhancements including real-time NSE WebSocket integration, LSTM-based predictive analytics, strategy backtesting, cross-platform mobile application development, and gamification features, the system is well-positioned for evolution into a comprehensive full-scale FinTech product capable of serving the rapidly growing retail investor population in India and other emerging markets.

REFERENCES

[1] S. Chandra and R. Mehta, "Digital Transformation in Indian Financial Markets," Journal of FinTech Research, vol. 4, no. 2, pp. 45–58, 2022.

[2] SEBI, "Investor Awareness and Financial Literacy in India," Securities and Exchange Board of India, New Delhi, 2023.

[3] Zerodha Varsity, "Paper Trading and Simulation in Modern Platforms," [Online]. Available: https://zerodha.com/varsity. [Accessed: Mar. 2025].

[4] R. Bauer, M. Cosemans, and P. Eichholtz, "Option Trading and Individual Investor Performance," Journal of Banking & Finance, vol. 33, no. 4, pp. 731–746, 2009.

[5] E. Chan, Algorithmic Trading: Winning Strategies and Their Rationale. Hoboken, NJ: Wiley, 2013.

[6] T. R. Reshef and O. Galor, "SaaS Business Models in Emerging FinTech Markets," International Journal of Financial Innovation, vol. 7, no. 1, pp. 12–29, 2021.

[7] Google Firebase, "Firebase Documentation – Authentication and Firestore," [Online]. Available: https://firebase.google.com/docs. [Accessed: Apr. 2025].

[8] React Dev, "React.js Official Documentation," [Online]. Available: https://react.dev. [Accessed: Apr. 2025].

[9] S. Raschka and V. Mirjalili, Python Machine Learning, 3rd ed. Birmingham, UK: Packt Publishing, 2019.

[10] Upstox Developer API, "Upstox API Documentation for Algorithmic Trading," [Online]. Available: https://upstox.com/developer/api-documentation. [Accessed: Mar. 2025].

[11] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[12] L. Breiman, "Random Forests," Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[13] NSE India, "National Stock Exchange of India – Market Data APIs," [Online]. Available:

https://www.nseindia.com/api. [Accessed: Apr. 2025].

[14] FastAPI, "FastAPI Documentation – Modern Web APIs with Python," [Online]. Available: https://fastapi.tiangolo.com. [Accessed: Apr. 2025].

[15] J. Waweru, "Behavioral Finance and Investor Decision Making in Emerging Markets," African Journal of Finance, vol. 12, no. 3, pp. 88–102, 2020.