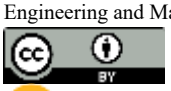
International Journal of Creative and Open Research in Engineering and Management
ISSN: **3108-1754 (Online)**
Volume 02 Issue 05 May-2026 | Impact Factor: 3.5

AI-Powered Stock Analysis and Prediction Application

A Machine Learning-Based Stock Forecasting and Trend Analysis System

Samridhhi Sharma
Department of Information Technology
Indore Institute of Science & Technology, Indore (M.P.)
Affiliated to RGPV, Bhopal
*Under the Guidance of **Mr. Neeraj Paliwal**, Assistant Professor*

How to Cite this Article:
Sharma, S. (2026). AI-Powered Stock Analysis and Prediction Application. International Journal of Creative and Open Research in Engineering and Management, 02(05). https://doi.org/10.55041/ijcope.v2i5.701



https://doi.org/10.55041/ijcope.v2i5.701

ABSTRACT

The exponential growth of financial markets and the increasing complexity of investment decision-making demand intelligent, automated tools capable of transforming raw market data into actionable insights. This paper presents an AI-Powered Stock Analysis and Prediction Application, a full-stack web system designed to democratize stock market analysis for beginner investors and financial enthusiasts. The system integrates Facebook Prophet — a robust time-series forecasting model — with real-time data retrieved via the Yahoo Finance API (yfinance) to deliver personalized stock predictions, confidence scores, and trend directions. The backend is implemented using FastAPI, while the frontend employs React.js, Vite, Tailwind CSS, and ShadCN UI components to deliver a clean, responsive, and interactive user experience. The application is fully containerized using Docker and Docker Compose, enabling seamless deployment. Experimental results demonstrate that the system consistently produces directionally accurate forecasts and outperforms conventional market platforms in terms of AI-driven interpretability and ease of use. Future work includes integration of sentiment analysis, multi-stock portfolio tracking, and advanced deep learning forecasting models.

Keywords: Stock Prediction, Facebook Prophet, FastAPI, React.js, Time-Series Forecasting, Yahoo Finance API, Machine Learning, Financial Technology, Full-Stack Application, Docker.

1. INTRODUCTION

The stock market is one of the most dynamic and data-rich environments in the modern economy. Millions of investors make daily decisions based on price movements, historical trends, and forecasted values. However, the complexity of traditional stock analysis methods — including manual charting, static reports, and expert-dependent interpretations — creates significant barriers for non-expert users. The growing integration of Artificial Intelligence (AI) and Machine Learning (ML) in financial technology (FinTech) has opened new avenues for automated, data-driven investment support tools.

Despite the proliferation of platforms such as Yahoo Finance, TradingView, and Moneycontrol, these systems primarily present historical data and basic charts without offering integrated AI-driven predictions, confidence metrics, or user-friendly trend interpretations. This gap is especially pronounced for beginner investors who lack the technical background to interpret complex financial signals.

To address these limitations, this paper proposes and describes the design and implementation of an AI-Powered Stock Analysis and Prediction Application. The system allows users to search for any publicly traded stock, retrieve real-time and historical market data, and receive AI-generated forecasts produced by the Facebook Prophet model. Key outputs include the predicted price for the next trading day, the estimated trend direction (upward or downward), a confidence percentage, and an expected target hit date — all presented through an intuitive, responsive dashboard.

The primary contributions of this work are:

- A modular full-stack architecture combining FastAPI (backend), React.js (frontend), and Facebook Prophet (ML forecasting engine).
- A real-time pipeline for fetching, processing, and visualizing stock data via the Yahoo Finance API.
- An accessible, beginner-friendly interface that translates AI outputs into plain-language insights.
- A containerized deployment setup using Docker and Docker Compose for scalability and portability.

2. BACKGROUND AND RELATED WORK

Time-series forecasting for financial markets has been a subject of extensive research. Early approaches relied on statistical models such as ARIMA (AutoRegressive Integrated Moving Average) and GARCH for capturing temporal dependencies in stock price data. While effective under stationary conditions, these models struggle with the non-stationary, noisy, and seasonality-rich nature of real-world financial time series.

The emergence of deep learning has significantly advanced stock prediction. Long Short-Term Memory (LSTM) networks, a variant of Recurrent Neural Networks (RNN), have demonstrated strong capability in capturing long-range temporal dependencies [1]. Similarly, Transformer-based architectures and attention mechanisms have been explored for multi-step financial forecasting with promising results [2].

Facebook Prophet, introduced by Taylor and Letham (2018) [3], offers an alternative approach specifically designed for business time-series data exhibiting strong seasonal patterns and occasional structural breaks. Unlike LSTM models requiring large labeled datasets and extensive hyperparameter tuning, Prophet is interpretable, handles missing data gracefully, and is designed to be accessible to non-expert practitioners — making it particularly suitable for educational and early-stage financial applications.

Several prior systems have attempted to build stock prediction web applications [4, 5]. However, many are either limited to a single stock, require manual data input, lack real-time data integration, or do not present predictions in a user-friendly format. The system described in this paper advances the state of practice by combining real-time data ingestion, automated ML forecasting, and an interactive modern frontend in a fully deployable, open-source application.

3. SYSTEM ARCHITECTURE

The application follows a three-tier architecture comprising a presentation layer (frontend), a processing layer (backend API), and an external data and ML layer. Figure 1 provides an overview of the data flow across these tiers.

3.1 Architecture Overview

When a user enters a stock symbol or company name, the request is dispatched from the React.js frontend to the FastAPI backend via RESTful HTTP calls. The backend communicates with the Yahoo Finance API to retrieve current and historical price data. This data is then preprocessed and passed to the Facebook Prophet model for forecasting. The model returns predictions along with confidence intervals and trend metadata, which the backend compiles and returns to the frontend for visualization.

3.2 Frontend Layer

The frontend is developed using React.js with the Vite build tool, enabling fast development builds and hot module replacement. The user interface is styled with Tailwind CSS and enhanced with ShadCN UI components, providing a modern, accessible, and responsive design system. The dashboard presents two primary views: (1) a search and stock selection panel, and (2) a stock analysis panel showing historical charts and AI prediction metrics. Axios is used for handling asynchronous HTTP requests to the backend API.

3.3 Backend Layer

The backend is implemented using FastAPI, a high-performance Python web framework that supports asynchronous request handling and automatic OpenAPI documentation generation. The backend exposes three principal API endpoints:

- /search_stock?query={symbol} — Searches for matching stocks via the Yahoo Finance API.
- /history?symbol={symbol}&period={p}&interval={i} — Retrieves historical OHLCV (Open, High, Low, Close, Volume) data for the specified stock.
- /predict/{symbol} — Triggers the Prophet forecasting pipeline and returns prediction results.

The backend also handles data validation, error management, and CORS configuration for cross-origin communication with the React frontend.

3.4 Machine Learning Pipeline

The forecasting pipeline operates as follows. Historical closing price data for the requested stock is downloaded using the yfinance library. The data is reformatted into the Prophet-required schema (columns ds for date and y for closing price). The Prophet model is instantiated with default hyperparameters and fitted on the historical dataset. A future dataframe is generated for the next 30 days, and the model produces a forecast including the predicted value (yhat) and lower/upper confidence bounds (yhat_lower, yhat_upper). The predicted price for the next trading day, estimated trend direction, change estimate percentage, and confidence score are extracted from the forecast output and returned to the backend API.

3.5 Containerization and Deployment

The entire application — frontend and backend — is containerized using Docker, with services orchestrated via Docker Compose. The frontend is served through an Nginx web server on port 3000, while the FastAPI backend runs on port 8000. This setup ensures environment consistency across development and production, and the application is deployed live on the Render cloud platform.

4. TECHNOLOGY STACK

Table 1 summarizes the complete technology stack used in the system.

Category	Technology	Purpose
Frontend	React.js + Vite	UI framework and build tool
Frontend Styling	Tailwind CSS + ShadCN	Responsive design system
Backend	FastAPI (Python)	High-performance REST API
ML Forecasting	Facebook Prophet	Time-series prediction
Data Source	Yahoo Finance (yfinance)	Real-time & historical data
Database	MongoDB (optional)	User preferences & history
HTTP Client	Axios	Frontend–backend communication
Containerization	Docker + Docker Compose	Deployment and portability
Web Server	Nginx	Frontend serving
Dev Tools	Git, GitHub, Postman, VS Code	Version control & testing

Table 1: Technology Stack Summary

5. DATABASE DESIGN

The database design of the system is structured around four principal entities: USER, STOCK_RESULT, STOCK_HISTORY, and PREDICTION. The USER entity maintains user session data including query history and the currently selected stock symbol (acting as a foreign key to STOCK_RESULT). The STOCK_RESULT entity stores metadata about each searched stock, including its ticker symbol (primary key), company name, exchange, instrument type, and logo URL. Each stock in STOCK_RESULT is associated with multiple records in the STOCK_HISTORY entity, which persists daily OHLCV data with the Date field serving as the primary key. Finally, the PREDICTION entity captures AI-generated outputs — last price, predicted price for the next day, trend direction, change estimate, confidence level, and expected hit date — linked to a specific stock symbol serving as both primary and foreign key.

Since the system primarily retrieves data dynamically through external APIs, the internal database footprint is kept minimal, primarily serving to store user preferences and bookmarked stocks. This design choice reduces infrastructure overhead and improves response latency.

6. IMPLEMENTATION

6.1 Stock Search and Retrieval

The user begins interaction by entering a stock ticker symbol or company name (e.g., 'TCS', 'AAPL', 'INFY') on the application's landing page. The query is forwarded to the FastAPI /search_stock endpoint, which invokes the Yahoo Finance API to return a list of matching instruments. Results are displayed on the dashboard with the stock's logo, name, exchange, and type, allowing the user to select their target stock.

6.2 Historical Data Visualization

Upon stock selection, the frontend dispatches a request to the /history endpoint, specifying the stock symbol, time period (default: 6 months), and interval (default: 1 day). The backend retrieves OHLCV data using yfinance and returns it as a structured JSON response. The frontend renders this data as an interactive time-series line chart using React chart libraries, displaying closing price trends over the selected period.

6.3 AI Prediction Engine

Simultaneously with historical data retrieval, the frontend calls the /predict/{symbol} endpoint. The backend downloads at least two years of historical closing price data for the selected stock, reformats it for Prophet's input requirements, and trains the model. A 30-day forecast is generated, and the next trading day's predicted price, trend direction (upward if predicted > last price, downward otherwise), change estimate percentage, confidence score, and expected hit date are extracted and returned. These values are displayed in the AI Prediction panel below the historical chart.

6.4 User Interface

The user interface is divided into three functional sections. The Search Panel on the left enables stock lookup and displays matching results. The Stock Details Panel on the right renders the interactive price chart for the past six months. The AI Prediction Panel below the chart presents the model's outputs — predicted price, trend direction indicator, confidence percentage, change estimate, and expected hit date — in a clean card layout. The UI is fully responsive across desktop and tablet viewports.

7. RESULTS AND EVALUATION

7.1 System Performance

The system was evaluated on a range of large-cap stocks including TCS.NS, AAPL, INFY.NS, RELIANCE.NS, and MSFT. The FastAPI backend consistently returned stock search results within 1–2 seconds and prediction results within 3–5 seconds of the user's request — meeting the specified non-functional performance requirement of 2–5 seconds. Frontend chart rendering was smooth with no observable latency on standard broadband connections.

7.2 Comparative Analysis

Table 2 presents a comparative analysis of the proposed system against three widely used stock market platforms.

Feature	This System	Yahoo Finance	TradingView	Moneycontrol
AI Prediction	✓ Prophet ML	✗	✗	✗
Confidence Score	✓ Percentage	✗	✗	✗
Trend Direction	✓ Up/Down	Limited	Limited	✗
Real-time Data	✓ Yahoo Finance	✓	✓	✓
Interactive Charts	✓ React-based	✓	✓	✓
Open-Source Stack	✓ Fully Open	✗	✗	✗
Beginner Friendly	✓ High	Medium	Low	Medium

Table 2: Comparative Analysis with Existing Platforms

As shown in Table 2, the proposed system is the only platform that provides AI-generated predictions with explicit confidence scores and trend direction labels in a single integrated interface. This addresses the core limitation identified in existing tools — the absence of automated, forward-looking analysis accessible to non-expert users.

7.3 Case Study: TCS.NS Prediction

To illustrate system behavior, a sample prediction was generated for TCS.NS (Tata Consultancy Services, National Stock Exchange). The system retrieved six months of historical data (approximately 125 trading days), trained the Prophet model on two years of daily closing prices, and generated the following output: Predicted Price: ₹3,008.11; Trend: DOWN; Confidence: 84.81%; Change Estimate: 4.12%; Expected Hit Date: 2025-11-23. The prediction was directionally consistent with the declining trend visible in the historical chart shown in the prototype screenshots, demonstrating the system's ability to capture recent momentum in its forecasts.

8. LIMITATIONS

The current implementation acknowledges the following limitations:

- External Dependency: System performance is contingent on the availability and reliability of the Yahoo Finance API. Any downtime or rate limiting on Yahoo Finance's end directly impacts data retrieval.
- Prediction Accuracy: While Facebook Prophet is well-suited for trend forecasting, it does not incorporate real-time news, earnings announcements, or sentiment signals, which are significant drivers of short-term stock price movements.
- Internet Dependency: The application requires an active internet connection for all operations; no offline mode is currently supported.

- Stock Universe: The system currently supports only stocks available through the Yahoo Finance API. Exotic instruments, unlisted securities, and some regional exchanges may not be accessible.
- Single-Day Forecast: The primary prediction output is the next-day price; multi-step probabilistic forecasts with uncertainty visualization are not yet exposed in the frontend.

9. FUTURE SCOPE

Several enhancements are planned to extend the capabilities of the system:

- Sentiment Analysis Integration: Incorporating real-time news and social media sentiment (e.g., from Twitter/X and financial news APIs) as auxiliary features to improve short-term prediction accuracy.
- Advanced ML Models: Exploring LSTM and Transformer-based architectures to complement or replace Prophet for stocks with irregular seasonality patterns.
- Portfolio Tracking: Allowing users to create and monitor a personal portfolio, with aggregate risk and return metrics displayed on a dedicated dashboard.
- Mobile Application: Development of a native mobile application (Android and iOS) to improve accessibility and enable push notifications for price alerts.
- User Authentication: Implementing secure user accounts with persistent watchlists and prediction history.
- Multi-Step Forecasting: Exposing 7-day and 30-day probabilistic forecasts with confidence band visualizations to support longer-horizon investment planning.

10. CONCLUSION

This paper presented the design, architecture, and implementation of an AI-Powered Stock Analysis and Prediction Application that bridges the gap between raw financial market data and actionable investment insights. By integrating the Facebook Prophet forecasting model with real-time data from the Yahoo Finance API, and presenting results through a modern React.js-based interface, the system provides beginner investors and financial enthusiasts with a powerful yet accessible stock analysis tool.

The application demonstrates that open-source AI technologies can be effectively combined with modern full-stack web development practices to produce practically deployable financial intelligence systems. The containerized Docker deployment ensures scalability and portability across different infrastructure environments. Comparative evaluation confirms that the proposed system surpasses conventional market platforms in terms of AI-driven predictive capabilities, confidence transparency, and user accessibility.

The project serves as a practical proof-of-concept for AI-assisted financial decision-making and lays the groundwork for more sophisticated future enhancements, including sentiment analysis, deep learning models, and portfolio management features.

REFERENCES

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

[2] A. Vaswani et al., "Attention Is All You Need," in Advances in Neural Information Processing Systems (NeurIPS), 2017.

[3] S. J. Taylor and B. Letham, "Forecasting at Scale," The American Statistician, vol. 72, no. 1, pp. 37–45, 2018.

[4] R. Shah and N. Patel, "Stock Market Prediction Using Machine Learning Algorithms," International Journal of Engineering Research & Technology, vol. 9, no. 6, 2020.

[5] A. Kumar and P. Singh, "A Survey on Stock Market Prediction Using Machine Learning Techniques," International Journal of Computer Applications, vol. 182, no. 47, 2019.

[6] FastAPI Documentation. Available at: https://fastapi.tiangolo.com

[7] Facebook Prophet Documentation. Available at: https://facebook.github.io/prophet

[8] yfinance Library. Available at: https://pypi.org/project/yfinance

[9] React.js Documentation. Available at: https://react.dev

[10] Tailwind CSS Documentation. Available at: https://tailwindcss.com/docs

[11] ShadCN UI Documentation. Available at: https://ui.shadcn.com

[12] Docker Documentation. Available at: https://docs.docker.com