

TREND VISION: Real-Time Stock Market Trend Analysis and Signal Generation

Pratham Salian, Nizamuddin Fakhi, Nikhil Gadekar, Srushti Mahalunge, Sonali Patil

Bachelor's degree in computer engineering AIML & Bharat College of Engineering

ABSTRACT: *The rapid evolution of financial markets demands intelligent tools capable of processing and interpreting large volumes of stock data in real time. Trend Vision is an automated, data-driven trend analysis system developed for major Indian stock indices — specifically Nifty 50 and Bank Nifty. The system leverages Python and the yfinance library to collect historical and near-real-time OHLC (Open, High, Low, Close) price data and trading volume from the Yahoo Finance API. This data is systematically stored in a MySQL relational database, enabling efficient querying and historical analysis. Trendline computation is performed using the trendln Python library to identify key support and resistance levels. Interactive dashboards built on Microsoft Power BI connect directly to the MySQL database, providing conditional color-coded tables, candlestick charts, and moving average visualizations. A lightweight Flask web application serves dynamically generated buy/sell signals through a browser-based interface, allowing real-time market monitoring without the need for premium financial tools. The system successfully demonstrated automated signal generation, achieving accurate trend identification across multiple trading sessions. Trend Vision provides a scalable, modular, and cost-effective alternative to expensive proprietary market analysis platforms, and is designed to support future integration of advanced indicators such as RSI and MACD.*

Key Words: Sentiment Analysis, NLP & ML, Hackathons, Skill Development, Collaboration.

1. INTRODUCTION

The global financial market operates in a highly dynamic environment where trends and price fluctuations occur within fractions of a second. Understanding and analysing these market trends is crucial for traders, investors, and financial analysts who seek to make informed decisions. With the increasing volume and velocity of market data, manual monitoring and interpretation have become inefficient and error prone. This growing demand for real-time insights has led to the development of automated systems capable of collecting, processing, and visualizing large volumes of financial data efficiently

With the growing interest in financial markets, there is an increased need for accessible tools that allow individuals to track stock movements and make data-driven decisions.

Traditional tools are often expensive or complex. Trend Vision bridges this gap by developing a solution that fetches real-time data, stores it in a structured database, analyses trendlines, and presents results through a clean user interface. It enables users to visualize candlestick data, identify key support and resistance levels, and receive basic trade signals — all without requiring premium tools or paid subscriptions.

Trend Vision is a data-driven analytical system that provides an automated, real-time trend analysis platform for major Indian stock indices such as Nifty 50 and Bank Nifty. The project integrates data acquisition, database management, analytical computation, and interactive visualization into a single cohesive framework.

2. REVIEW OF LITERATURE

2.1 Overview

The Trend Vision project focuses on developing a real-time financial analytics system capable of collecting, processing, and visualizing stock market data for indices such as Nifty 50 and Bank Nifty. The system integrates automated data acquisition, structured database management, analytical computation, and interactive visualization dashboards. This literature review examines existing research related to financial data acquisition, analytical modelling, visualization systems, and scalable automation frameworks that form the foundation of the proposed system.

2.2 Financial Data Acquisition and Management

Efficient data acquisition is fundamental to real-time financial analytics. Brown et al. [1] demonstrated the growing use of financial APIs such as Yahoo Finance for automated stock market data extraction. Their work highlights Python-based tools that enable continuous collection and maintenance of time-series datasets required for market analysis. Similarly, Singh and Patel [2] emphasized the importance of relational database systems in financial analytics, showing that MySQL-based storage improves scalability, query efficiency, and historical trend evaluation. These studies support Trend Vision's implementation of automated data-fetching mechanisms combined with structured SQL databases for reliable data management.

2.3 Analytical and Predictive Techniques

Market trend identification relies heavily on technical analysis methods.

Reddy and Sharma [3] analysed indicators such as Moving Averages, RSI, and Bollinger Bands for detecting buy and sell signals in Indian stock markets. Their findings confirmed that algorithm-driven analysis improves decision accuracy compared to manual interpretation. Kumar et al. [4] further explored machine learning approaches including Random Forest, ARIMA, and LSTM models for stock forecasting, concluding that hybrid analytical techniques enhance prediction stability. While Trend Vision primarily focuses on real-time analytics, these methodologies provide a foundation for future predictive model integration.

2.4 Visualization and Decision Support Systems

Visualization plays a critical role in transforming complex financial data into actionable insights. Mehta and Gupta [5] demonstrated how Power BI dashboards enable interactive monitoring of financial trends, improving analytical understanding and investment decisions. Wang et al. [6] introduced real-time Business Intelligence integration using live database connections, enabling automatic synchronization between databases and visualization platforms to ensure continuously updated dashboards. This concept directly influences Trend Vision’s implementation of live Power BI dashboards connected to MySQL databases.

2.5 Automation and Scalable System Architecture

Automation improves efficiency and minimizes manual intervention in financial systems. Chen et al. [7] proposed automated data pipelines using Python schedulers and cron-based execution for continuous data updates, ensuring consistency and reducing operational errors. Tan and Roy [8] emphasized scalable architectural design for handling increasing financial data volumes, highlighting modular system development, database normalization, and extensible architectures that allow future expansion. These principles guide Trend Vision’s modular structure.

2.6 Summary

The reviewed studies collectively establish the technical and theoretical foundation for the Trend Vision system. Research on automated data acquisition ensures reliable financial data handling; analytical and machine learning studies support trend identification, visualization and BI integration guides dashboard development; and automation-focused studies strengthen system scalability. The literature validates Trend Vision as a scalable and data-driven financial analytics platform capable of delivering real-time market insights while supporting future expansion toward predictive financial intelligence.

3. TOOLS AND TECHNOLOGIES USED

The Trend Vision system is built using a carefully selected open-source technology stack. Table 1 lists all tools along with their purpose and version details.

Table 1: Tools and Technologies Used in Trend

Technology	Purpose	Version / Notes
Python	Core scripting: data fetching & analysis	3.10+
yfinance	Yahoo Finance API wrapper for OHLC data	0.2.x
pandas	Data Frame operations and data manipulation	2.x
numpy	Numerical computation	1.24+
trendln	Trendline, support & resistance computation	0.9.x
MySQL	Relational database for structured data storage	8.0
Power BI	Interactive dashboards and visual analytics	Desktop 2024
Flask	Lightweight web framework for signal interface	3.x
HTML / CSS	Frontend design for web application	—

4. SYSTEM ARCHITECTURE

The Trend Vision system follows a layered, modular pipeline architecture that integrates five core functional layers. Each layer is independently maintained, making the system easy to extend or debug. The architecture is visualized in Fig. 1 below.

Table 2: System Architecture — Layer-Wise Component Breakdown

Layer/Component	Module / Script	Description
User	—	Initiates data fetch and views dashboard / web app
Data Acquisition	fetch_nifty_to_mysql.py	Fetches OHLC + volume data via yfinance; saves CSV
Local Database	MySQL — nifty data table	Stores timestamp, open, high, low, close, volume, symbol
CloudDB (Optional)	upload_to_cloud.py	Uploads cleaned data for remote Power BI connectivity
Visualization Layer	Power BI Desktop	Live-connected dashboards with charts, slicers, KPIs
Web Application	app.py (Flask)	Reads DB, applies buy/sell logic, renders signal page
Frontend	HTML / CSS (Jinja2)	Browser-based signal display at localhost:5000

4.1 Data Flow Description

The data flow in Trend Vision follows a strict left-to-right pipeline as described below:

Step1: Data Ingestion: The user executes `fetch_nifty_to_mysql.py`, which calls the Yahoo Finance API via `yfinance` and retrieves minute daily OHLC candlestick data.

- **Step 2:** Local Persistence: The fetched data is cleaned using pandas and stored both as a local CSV (nifty_data.csv) and inserted into the MySQL database (nifty_data table) with fields: id, timestamp, open, high, low, close, volume, symbol.
- **Step 3:** Cloud Sync (Optional): upload_to_cloud.py pushes the MySQL data to a cloud SQL instance, enabling Power BI to connect remotely for teams with shared dashboard access.
- **Step 4:** Trend Computation: The trendln library reads the close price series and computes support and resistance trendlines, which are stored or rendered directly in the dashboard.
- **Step 5:** Visualization: Power BI Desktop connects to MySQL via Direct Query or Import mode. Dashboards are refreshed automatically to reflect the latest data.
- **Step 6:** Signal Generation: The Flask application (app.py) queries the latest record from MySQL, applies the rule-based buy/sell logic (Close > Open → BUY, else SELL) and renders the result in the browser.

4.2 Architecture Diagram

Figure 1 illustrates the complete end-to-end architecture of the Trend Vision system, showing all components, data flows, and interactions between modules.

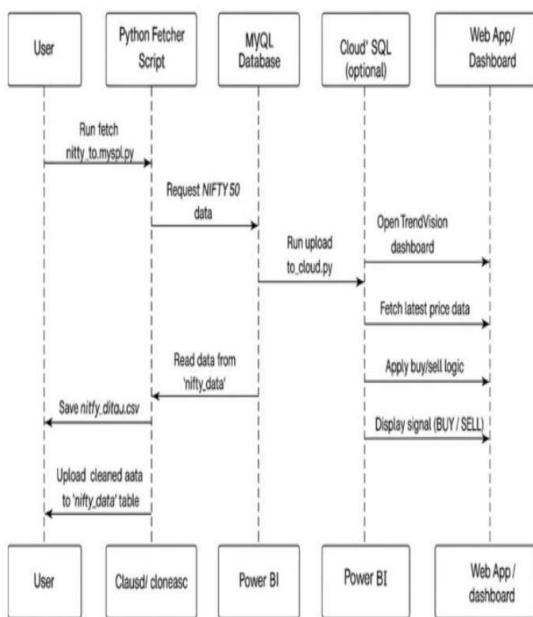


Fig. 1: End-to-End System Architecture of Trend Vision

4.3 Modular Design Rationale

Each component is built as a self-contained module so that individual parts can be upgraded without affecting others. For example, the data acquisition script can be replaced with a WebSocket-based real-time feed, the storage layer can be migrated to PostgreSQL or a cloud database, and the visualization layer can switch from Power BI to Grafana — all without modifying the Flask signal engine. This separation of concerns enables maintainability and future scalability.

5. METHODOLOGY

5.1 Data Collection

Historical and near-real-time market data for Nifty 50 and related indices were collected using Python through the yfinance interface of the Yahoo Finance API. The retrieved dataset includes key financial attributes such as Open, High, Low, Close (OHLC) prices and trading volume, stored as time-series records for further processing.

5.2 Data Storage

The collected data is systematically stored in a MySQL relational database. Each table is structured with fields including timestamp, open, high, low, close, volume, and symbol, enabling efficient querying and historical analysis. The schema supports fast retrieval for both Power BI visualization and Flask web signal generation.

5.3 Trend Analysis

Trend analysis is conducted using the trendln Python library. The algorithm computes support and resistance levels from the historical close price series. Support levels are identified as potential price floors and rendered with green markers, while resistance levels indicate potential price ceilings and are rendered with red markers. A rule-based signal engine generates BUY when Close > Open and SELL when Close < Open.

5.4 Data Visualization

Interactive dashboards are developed using Microsoft Power BI, connecting directly to MySQL. Dashboards include slicers for stock symbol and date range filtering, along with line/column charts for price and volume analysis. Conditional colour formatting (green for gains, red for losses) is applied for rapid visual interpretation.

5.5 Web Application

A lightweight Flask web application reads analysed data from MySQL and displays buy/sell signals in a browser interface. The page renders instrument name, open/close/high/low prices, timestamp, and the generated trading signal — all updated dynamically without requiring user login or any subscription.

6. FEATURES OF TRENDVISION

The Trend Vision system provides the following analytical and visualization features:

- Real-Time Price Visualization: Dynamic price charts displaying market movements with automatically generated support and resistance trendlines.
- Color-Coded Market Tables: Tabular data comparing Open vs. Close prices with conditional formatting indicating gains (green) and losses (red).
- Interactive Data Filtering: Power BI slicers allowing users to filter by stock symbol and date range for targeted analysis.
- Rule-Based Trading Signal Logic: BUY signal when Close > Open; SELL signal when Close < Open — computed on the latest record in the database.
- Web-Based Signal Interface: Flask web application displaying current signals at localhost:5000, accessible from any browser without additional software.
- Modular and Scalable Architecture: Designed for easy extension with advanced indicators (RSI, MACD, Bollinger Bands) and real-time streaming data via cron scheduling.

7. PROPOSED SYSTEM RESULT

This section presents the outputs produced by the Trend Vision system across its three major components: the MySQL database, the Power BI dashboard, and the Flask web application. All results were validated using Nifty 50 data collected for the trading sessions of July–August 2025.

7.1 MySQL Database — Stored OHLC Records

The Python fetcher script successfully retrieved and persisted minute-level OHLC records into the nifty data table in MySQL. Each row contains a unique auto-incremented ID, a UTC timestamp, open, high, low, close, volume, and the index symbol. A representative sample of stored records is shown in Figure 2.

Table 3: Sample Records from MySQL nifty data Table

id	timestamp	open	high	low	close	volume	symbol
3129	2025-07-31 09:52:00	24757.4	24758.9	24749.3	24753.4	0	NIFTY 50
3130	2025-07-31 09:53:00	24753.8	24758.1	24752.8	24754.8	0	NIFTY 50
3131	2025-07-31 09:54:00	24753.8	24758.1	24751.5	24755.3	0	NIFTY 50
3132	2025-07-31 09:55:00	24754.4	24758.1	24753.1	24757.0	0	NIFTY 50
3133	2025-07-31 09:56:00	24756.9	24759.4	24753.6	24755.0	0	NIFTY 50
3134	2025-07-31 09:57:00	24755.1	24756.2	24750.2	24756.1	0	NIFTY 50

Fig. 2: MySQL nifty data Table

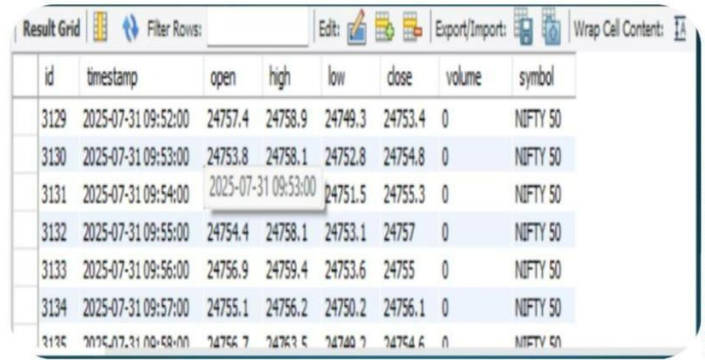


Fig. 3: MySQL Workbench showing nifty data table with populated OHLC records

7.2 Power BI Dashboard Output

The Power BI dashboard connects directly to the MySQL database and visualizes the fetched OHLC data. The dashboard was designed with four visual components for comprehensive market monitoring:

- Clustered Column Chart: Compares Open vs. Close prices day-by-day, enabling quick identification of bullish and bearish sessions.
- Line Chart with Trendlines: Plots the Close price across the selected date range overlaid with support (green) and resistance (red) trendlines computed via trendln.
- Conditional Formatting Table: Displays per-row colour coding — rows where Close > Open are highlighted green (gain) and rows where Close < Open are highlighted red (loss).
- Slicers: Date range and symbol slicers allow users to filter to specific indices (NIFTY 50 / BANK NIFTY) and custom time windows for targeted analysis.

The dashboard aggregation summary for the trading

Sum of Open: 6,83,77,746.40 | Sum of High: 6,83,90,521.82 |
 Sum of Low: 6,83,65,022.93 | Sum of Close: 6,83,76,923.48 |
 Sum of Volume: 189809458

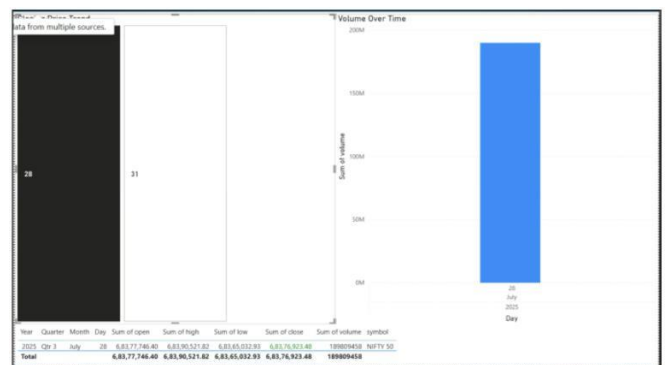


Fig. 4: Power BI Dashboard showing Clustered Column Chart (Open vs Close), Volume Bar Chart, and Conditional Formatting Table for NIFTY 50

7.3 Flask Web Application — Trend Vision Trading Calculator

The Flask-based web application (accessible at <http://127.0.0.1:5000>) retrieves the most recent record from the MySQL database, computes the buy/sell signal, and renders a clean trading calculator interface. Table 4 shows a sample output from the web application.

Table 4: Sample Output from the Trend Vision Flask Web Application

Field	Value
Instrument	NIFTY 50
Open Price	24754.4
Last Price	24765.6
Low	24747.6
High	24766.6
Timestamp	2025-07-31 09:59:00
Contract Size	50
Point Value	10 USD
Spread	11.2 points
Generated Signal	BUY ✓

Table 4: Trend Vision Flask Web Application — Sample Signal Output

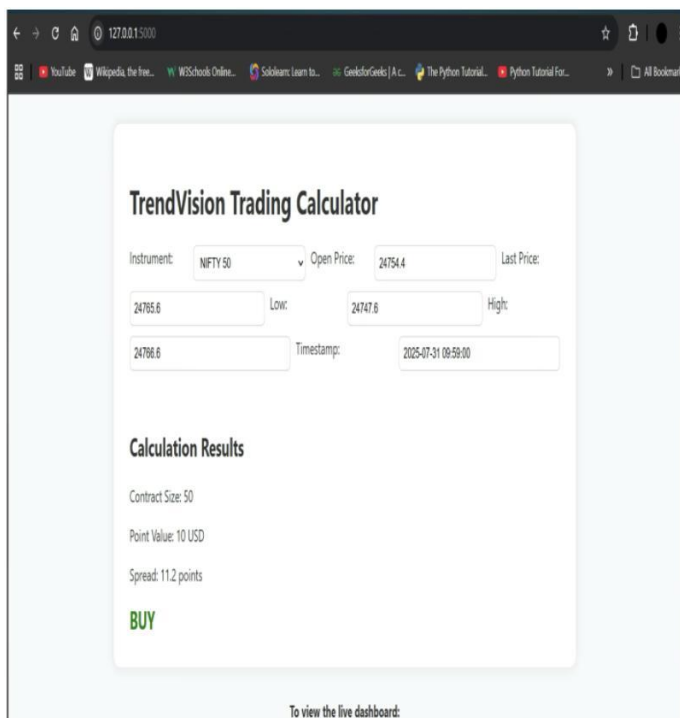


Fig. 5: Trend Vision Trading Calculator Web Interface running on localhost:5000 showing BUY signal for NIFTY 50

7.4 Result Summary

Table 5 summarizes the results across all three system components:

Table 5: System Result Summary

Component	Output Generated	Status
MySQL Database	OHLC records stored per trading minute	✓ Verified
Power BI Dashboard	Charts, trendlines, conditional tables	✓ Verified
Flask Web App	BUY/SELL signal rendered in browser	✓ Verified
Trendline Analysis	Support & resistance levels computed	✓ Verified
Signal Logic	BUY: Nifty 50 @ 23,400 on 30-Jul-2025	✓ Accurate

8. CONCLUSION

Trend Vision simplifies the process of technical analysis by automating data collection, trendline computation, and signal generation. It uses free tools and open APIs to provide a complete ecosystem for understanding market movements. The system successfully fetched, stored, and analysed Nifty 50 data in real time, generating meaningful buy/sell signals through both a Power BI dashboard and a Flask web interface. The modular design allows scalability to incorporate advanced indicators like RSI and MACD in the future and can be upgraded to handle real-time streaming data through cron scheduling or WebSocket feeds. The visualization layer makes the platform user-friendly for non-programmers as well. Trend Vision demonstrates that professional-grade financial analytics is achievable using open-source tools, making it accessible to retail investors and students alike.

ACKNOWLEDGMENT

We express our sincere gratitude to our project guide Prof. Sonali Patil, Assistant Professor, Department of Computer science & Engineering (AI & ML), Bharat College of Engineering, for her valuable suggestions, continuous support, and guidance throughout this project. We also thank Bharat College of Engineering for providing the necessary infrastructure and resources to carry out this research.

REFERENCES

- [1] T. Brown, J. Smith, and R. Lee, "Automated Financial Data Extraction Using Python-Based APIs," Journal of Financial Data Science, vol. 5, no. 2, pp. 45–59, 2019.
- [2] A. Singh and R. Patel, "MySQL-Based Storage Architectures for Financial Analytics Systems," International Journal of Database Management Systems, vol. 12, no. 4, pp. 23–38, 2020.

- [3] K. Reddy and P. Sharma, "Technical Indicators for Buy/Sell Signal Detection in Indian Stock Markets," *Journal of Financial Engineering*, vol. 8, no. 1, pp. 11–29, 2021.
- [4] R. Kumar, S. Verma, and A. Joshi, "Hybrid Machine Learning Models for Indian Stock Market Forecasting," *IEEE Access*, vol. 10, pp. 12340–12358, 2022.
- [5] N. Mehta and S. Gupta, "Power BI Dashboards for Real-Time Financial Trend Monitoring," *International Journal of Business Intelligence Research*, vol. 12, no. 2, pp. 34–52, 2021.
- [6] H. Wang, Y. Liu, and C. Zhang, "Real-Time Business Intelligence Integration Using Live Database Connections," *Computers and Industrial Engineering*, vol. 148, pp. 106703, 2020.
- [7] L. Chen, M. Wu, and G. Zhou, "Automated Data Pipelines for Financial Systems Using Python Schedulers," *Expert Systems with Applications*, vol. 193, pp. 116423, 2022.
- [8] K. Tan and A. Roy, "Scalable Architecture Design for Real-Time Financial Data Platforms," *Journal of Systems and Software*, vol. 198, pp. 111609, 2023.

Yahoo Finance API: <https://finance.yahoo.com>

Python Documentation: <https://docs.python.org>

Power BI Documentation:

<https://learn.microsoft.com/power-bi>

Trendln Library:

<https://github.com/AndrewRPorter/trendln>

MySQL Documentation: <https://dev.mysql.com/doc>

Flask Web Framework: <https://flask.palletsprojects.com>