A

Synopsis on

**TradeSense – Smart Trading Assistant**

*in partial fulfillment of the requirement for the degree*

of

Bachelor of Technology In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

**Drishay Chauhan (2301330100084)**

Under the supervision of

**Mr. Rohit Chaudhay** (Assistant Professor)

**Mrs. Anamika** (Assistant Professor)



# NOIDA INSTITUTE OF ENGINEERING AND TECHNOLOGY GREATER NOIDA

**Index**

|  |  |  |
| --- | --- | --- |
| **Sr.No.** | **Topics** | **Page No.** |
| 1. | Introduction | 3 |
| 2. | Existing Systems | 4 |
| 3. | Problem Statement | 5 |
| 4. | Proposed Methodology | 6-7 |
| 5. | Feasibility Study | 8 |
| 6. | Facilities required for proposed work | 9 |
| 7. | Conclusion | 10 |
| 8. | References | 11 |

Supervisor Sign:

**Introduction**

Retail participation in equity and crypto markets has grown exponentially over the past decade. India alone now hosts over 154 million active demat accounts, while global cryptocurrency trading attracts millions more. Despite this surge, most individual traders still depend on manual chart analysis and subjective judgment, resulting in inconsistent performance and missed opportunities.  
  
TradeSense addresses these challenges by delivering a cross-market, Python-based desktop application that:  
• Automates technical analysis across Indian equities, US stocks, and cryptocurrencies.  
• Generates real-time buy/sell signals via customizable Simple Moving Average (SMA) crossovers.  
• Incorporates risk management—stop-loss recommendations and position-sizing hints.  
• Logs every decision into an Excel workbook for easy review and audit.  
  
Field of the Project:  
Financial Technology (FinTech), Data Analytics, Algorithmic Trading.  
  
Special Technical Terms:  
• OHLCV: Open-High-Low-Close-Volume market data.  
• SMA Crossover: A strategy where buy/sell signals are generated when a short-term moving average crosses a long-term moving average.  
• Stop-Loss: Predetermined price at which a losing trade is automatically closed to limit loss.  
• Position Sizing: Determining the amount of capital allocated to each trade.  
• Modular Architecture: Clean separation of configuration, data ingestion, indicator logic, logging, and UI.

## ****Existing Systems****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| System Name | Type | Key Features | Limitations | Used By / Region |
| Zerodha Kite | Web/Mobile Trading Platform | Charting, basic alerts, order placement | No built-in risk controls; manual indicator setup | India |
| Groww | Web/Mobile Trading App | User-friendly UI, watchlists, basic technicals | Limited advanced indicators; no customizable strategy alerts | India |
| TradingView | Web-based Charting & Alerts | Extensive indicator library, Pine Script custom alerts | Paid subscription for real-time data; no integrated risk module | Global |
| Upstox Pro | Web/Mobile Trading Platform | Advanced charts, conditional orders, margin calculators | Lacks automated signal generation; no cross-market support | India |
| Crypto Exchanges | Web/Mobile (Binance, KuCoin) | Real-time market data, API access for algo-trading | Varying API limits; risk management left to user | Global (crypto) |

While these platforms excel in charting and basic alerts, they rarely bundle fully automated signal engines with built-in risk parameters or cross-asset support in a free, desktop-based solution.

**Problem Statement**

Retail traders face three major challenges:

1. **Time and Complexity:** Monitoring multiple assets, timeframes, and indicators (like moving averages or RSI) is labor-intensive and error-prone.
2. **Emotional Decision-Making:** Fear of missing out (FOMO) or panic-selling in downturns often overrides systematic strategy.
3. **Fragmented Tools:** While platforms like TradingView or Zerodha offer alerts, they seldom bundle them with customizable risk controls or cross-market support, and they may require paid subscriptions.

These limitations leave many traders struggling to execute consistent, disciplined strategies.

.

## ****Proposed Methodology****

## 4.1 System Architecture & Modules

• Configuration Module: Stores user-set parameters: tickers, SMA periods, capital allocation rules.

• Data Ingestion Module: Fetches live OHLCV data (yfinance for stocks; exchange APIs for crypto). Maintains rolling windows for short-term (e.g. 20-period) and long-term (e.g. 50-period) SMAs.

• Signal Engine: Detects SMA crossovers: Buy when short SMA crosses above long SMA; Sell when short SMA crosses below long SMA. Tags each signal with stop-loss and position-size suggestion.

• Risk Management Layer: Applies default and user-customizable risk rules. Validates suggested trade sizes against available capital.

• Visualization & GUI: Tkinter desktop app embeds Matplotlib charts. Sidebar shows latest signal, SMA settings, and ticker selector.

• Logging & Audit: Records timestamp, symbol, price, SMA values, and signal reason to TradeSense\_Signals.xlsx via openpyxl.

• Extension Hooks: Future: AI-driven indicators, automated order execution, mobile/web widgets.

## 4.2 Diagrams & Use Cases

Use-Case Diagram (Placeholder)  
Actors: Retail Trader, TradeSense App — Select Ticker → Analyze → View Signal → Log Signal → (Manual) Order Execution  
  
ER Diagram (Placeholder)  
Entities: User, Ticker\_Settings, Signal\_Log  
  
Data Flow Diagram (Level 0) (Placeholder)  
User Input → Data Fetcher → Signal Engine → GUI & Logging  
  
Flowchart (Signal Generation) (Placeholder)  
1. Fetch new data → 2. Calculate SMAs → 3. Crossover? → 4a. Generate/Log Signal → 4b. Update GUI

## 4.3 Development Methodology

An Iterative and Incremental model is adopted:  
• Iteration 1: Core SMA signal engine + basic GUI  
• Iteration 2: Risk management integration + Excel logging  
• Iteration 3: Polishing UI, adding multi-asset support  
• Iteration 4: Beta testing & feedback → refinements

## ****Feasibility Study****

|  |  |
| --- | --- |
| Feasibility Type | Assessment |
| Technical | All required libraries (yfinance, pandas, matplotlib, tkinter, openpyxl) are open-source. Python 3.8+ is widely available. |
| Operational | Desktop app runs on Windows/macOS/Linux; minimal user training; modular design eases maintenance. |
| Economic | Zero licensing costs; uses free APIs; development on standard hardware; minimal hosting if data caching is required. |
| Legal & Ethical | No sensitive personal data; all market data is public; users must acknowledge “no financial advice” disclaimer. |

# Facilities Required for Proposed Work

## 6.1 Software

• OS: Windows 10/11, macOS, or Linux  
• Language: Python 3.8+  
• IDE: VS Code / PyCharm  
• Libraries:  
 - Data: yfinance, pandas  
 - Visualization: matplotlib, tkinter  
 - Logging: openpyxl  
• Version Control: Git / GitHub

## 6.2 Hardware

• Processor: Intel i3 / AMD Ryzen 3 or better  
• RAM: ≥ 4 GB (8 GB recommended)  
• Storage: ≥ 256 GB HDD/SSD  
• Internet: Required for live data fetching

**Conclusion**

Upon completion of Phase 1, TradeSense will deliver a functional desktop application capable of:

* Fetching real-time data for diverse assets.
* Generating and displaying SMA-based buy/sell signals.
* Safeguarding trader capital through built-in risk parameters.
* Logging decisions for performance analysis.

This MVP not only simplifies technical analysis for retail traders but also establishes a robust, modular codebase ready for advanced enhancements—such as AI-driven indicators, automated order execution, and multi-market expansions.

**References**

1. **Yahoo Finance (yfinance) Documentation**
2. **Investopedia** – Articles on SMA and crossover strategies
3. **Python.org** – Tkinter and Matplotlib user guides
4. **openpyxl** – Excel file handling in Python
5. Relevant GitHub projects and Medium tutorials on Python trading bots