**Project Synopsis: TradeSense – Smart Trading Assistant**

**1. Project Title**

**TradeSense** – A cross-market, Python-based trading assistant that delivers real-time buy and sell signals alongside built-in risk management.

**2. Introduction**

Over the last decade, retail participation in financial markets has exploded. In India alone, there are over **154 million** active demat accounts, and globally, cryptocurrencies have brought trading to millions more. Yet most individual investors still rely on manual chart reading and gut feel, leading to missed opportunities and erratic performance. TradeSense aims to bridge this gap by automating technical analysis across **Indian equities, US stocks, and cryptocurrencies**, presenting clear, data-driven signals in an accessible desktop application.

**3. Problem Definition**

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.

**4. Objectives**

TradeSense is designed to:

* **Automate Market Analysis:** Seamlessly fetch live OHLCV data via free public APIs (yfinance) for stocks and crypto.
* **Generate Actionable Signals:** Compute customizable short- and long-period Simple Moving Averages (SMAs) and detect crossovers to trigger buy or sell alerts.
* **Embed Risk Management:** Apply default stop-loss and position-sizing parameters to each signal, with future provisions for user adjustment.
* **Deliver via Intuitive GUI:** Show price charts, overlay SMAs, display the latest signal status, and log every decision to an Excel file for audit and review.
* **Maintain Modular Design:** Lay a code foundation that cleanly separates configuration, data ingestion, indicator logic, logging, and user interface—enabling rapid extension to AI models, mobile widgets, or web integrations in later phases.

**5. Proposed Methodology**

1. **Data Ingestion:** Use yfinance to pull recent price and volume data. For each selected ticker, the system maintains a short-term (e.g. 20-period) and long-term (e.g. 50-period) SMA.
2. **Signal Engine:** On each updated data point, the engine checks for SMA crossovers—when the short average crosses above the long, a **Buy** signal is generated; when it crosses below, a **Sell** signal is issued.
3. **Risk Management Layer:** Each signal is tagged with a recommended stop-loss percentage (e.g. 2% below entry) and a position-size suggestion based on predefined capital allocation rules.
4. **Visualization & Interaction:** A **Tkinter** desktop application presents an embedded **Matplotlib** chart showing the closing price and both SMAs. The most recent signal (“BUY” or “SELL”) is highlighted in the sidebar, alongside controls to select tickers and SMA periods.
5. **Logging:** All signals—including timestamp, symbol, price, SMA values, and reason—are recorded to an Excel workbook (TradeSense\_Signals.xlsx) using openpyxl, allowing users to review performance and refine parameters.

**6. Flowchart & Use-Case**

* **Flowchart Steps:**
  1. User selects market and ticker.
  2. Application fetches latest data.
  3. Indicator engine calculates SMAs and checks for crossovers.
  4. If a crossover occurs, signal is displayed and logged.
  5. Trader reviews signal and (manually) places orders in their brokerage account.
* **Primary Use-Case:**
  1. **Actor:** Retail trader seeking systematic guidance.
  2. **Scenario:** Trader launches TradeSense, selects “INFY.NS,” clicks “Analyze,” and immediately sees a “BUY” alert with a chart and default risk settings, enabling prompt action.

**7. Feasibility Study**

* **Technical Feasibility:** All core libraries (yfinance, pandas, matplotlib, tkinter, openpyxl) are open-source and freely available. No proprietary data feeds or paid services are required for Phase 1.
* **Economic Feasibility:** Minimal cost—development uses existing hardware and free APIs. Operating cost is limited to standard internet connectivity.
* **Operational Feasibility:** The desktop GUI is lightweight, runs on Windows/macOS/Linux with Python installed, and requires no complex deployment. Retail traders can install and use the tool without specialized knowledge.

**8. Hardware / Software Requirements**

* **Hardware:** Any modern PC or laptop with at least 4 GB RAM and an internet connection.
* **Software:**
  + **OS:** Windows 10/11, macOS, or most Linux distros.
  + **Python:** Version 3.8 or later.
  + **Libraries:** Installed via pip install yfinance pandas matplotlib openpyxl.

**9. Expected Results & 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.

**10. 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