Based on your requirements and the available market data, I will design two distinct automated trading logics for Nifty 50 options, focusing on high-probability setups and capital protection for your ¹ 1 0 , 0 0 0 account. The core philosophy for both strategies is to use **multi-layered confirmation** to increase the probability of success, with an exit plan predetermined before any trade is entered.

The table below summarizes the two strategy archetypes.

Strategy Aspect	**Logic A: Quick Scalping**	**Logic B: High Conviction**
Core Objective	Capture small, frequent profits from minor price moves.	Capture larger moves by waiting for strong, aligned market signals.
Holding Period	Seconds to a few minutes.	15 minutes to a few hours.
Trade Frequency	High (multiple trades per session).	Low (1-3 high-quality trades per day).
Primary Data Focus	Ultra-fast momentum, order flow (VWAP), and short-term Greeks (Gamma).	Multi-timeframe trend alignment, volatility context (India VIX), and sentiment (PCR/OI).
Ideal Market	Consistent, moderate volatility (not extreme).	Clear trending or high-volatility breakout environments.

Ø>Ýà Core Logic for Entries and Market Filters

A robust bot acts like a team of specialists. The first agent, the **Signal Agent**, should analyze market state and generate signals.

1. Market State Filter: Trending vs. Sideways

Before any trade, the bot must classify the market to avoid false signals in choppy conditions.

*Method**: Implement a consolidated score using multiple dimensions. A simple but effective filter for a 1-second environment can be built using:

*Price vs. VWAP & EMAs**: Price consistently above/below the intraday VWAP and a fast EMA (e.g., 21-period) indicates a trend. If price oscillates around these levels, the market is sideways.

*ATR & Range Detection: Calculate the Average True Range (ATR) over 2000 bars. If the price remains within a band defined by `(Weighted Average Price ± (ATR* Multiplier))` for a sustained period (e.g., 50-100 bars), label the market as "Sideways/Ranging".

*Python Logic Suggestion**:

```python
# Pseudo-code for market state function
def assess*market*state(price, vwap, ema_fast, atr):
trend_strength = 0
# Check alignment
if price > vwap and price > ema_fast:
```

```
trend_strength += 1 # Bullish bias
elif price < vwap and price < ema_fast:
trend_strength -= 1 # Bearish bias

# Check if in a tight range (Sideways)
recent_high = max(price[-50:])
recent_low = min(price[-50:])
rangewidthpct = (recenthigh - recentlow) / recent_low

if rangewidthpct < 0.005 and abs(trend_strength) < 2: # 0.5% range
return "SIDEWAYS"
elif trend_strength >= 2:
return "UPTREND"
elif trend_strength <= -2:
return "DOWNTREND"
else:
return "CHOPPY"
```

2. Entry Logic Specifications

_Logic A (Quick Scalping)_*: Aim for precision on a 1-minute chart.
_Primary Trigger: Look for a Momentum Break_* from a key level. This could be a break above/below the volume-weighted average price (VWAP) or a short-term high/low.
_Confirmation (Must have all)_*:
1. **Gamma & Delta Surge**: Monitor ATM option Gamma for acceleration. A rising Delta (e.g., moving above 0.55 for calls or below -0.45 for puts) confirms directional probability is increasing.
2. **Micro-Momentum**: A fast stochastic (e.g., %K 5,3) crossing in the direction of the trade, supported by a 1-second RSI(6) moving sharply from below 40 to above 60 (for calls) or vice versa.
_Direction_*: Only trade in the direction of the immediate trend identified by the market state filter.

_Logic B (High Conviction)_*: Wait for a "stacked" signal on a 5-minute chart.
_Primary Trigger: A Breakout_* from the first 15-minute high/low range of the trading day is a powerful intraday signal.
_Confirmation (Must have all)_*:
1. **Trend Alignment**: Price must be above VWAP and a fast EMA (e.g., 21) for a call, or below for a put.
2. **Sentiment Alignment**: Put-Call Ratio (PCR) should support the move (e.g., falling PCR for a call breakout, rising PCR for a put breakout).

3. **Volatility Context**: Check India VIX. A rising VIX can fuel a strong trending move, but an extremely high VIX (e.g., >20) may signal excessive choppiness.
*Direction: The bot should have a manually set or algorithmically determined daily bias** (Bullish/Bearish) and only take signals that align with it.

&–þRisk & Profit Protection Parameters
This is managed by the **Position Management Agent**. Rules must be absolute and mechanical.

*Optimal Stop Loss**:
*For Logic A (Scalping): Use a dynamic stop** based on the option's real-time Delta. For example, calculate the spot price move that would cause a 40-50% loss in the option's premium and set that as the index point stop. Given the speed, this is often a tight stop (5-10 Nifty points).
*For Logic B (High Conviction): Use a wider, fixed stop at a key structure level. This could be the midpoint of the first 15-minute range or a recent swing low/high. As a rule, initial risk should not exceed 1-2% of your ¹ 1 0 , 0 0 0 capital** per trade.

*Profit Protection (Trailing): For both logics, implement a trailing stop order**.
* For scalping, trail by a fixed monetary value (e.g., ¹ 3 0 profit per lot) or a tight moving average of the option's price.
* For high-conviction trades, use a more sophisticated trail, such as anchoring to the VWAP or a parabolic SAR. The goal is to let profits run while the trend is intact but exit sharply on reversal.

*Time Decay (Theta) & Stagnation Rule**: Time decay (theta) is the enemy of an option buyer.
*Absolute Rule: Do not enter a long option trade within the last 60 minutes** of the trading day. Theta decay accelerates dramatically.
*Stagnation Rule: If a trade is open and the price has not moved in your favor by more than the bid-ask spread within 90 seconds (for Logic A) or 5 minutes (for Logic B)**, exit immediately. This protects capital from slow theta bleed.

Ø=ÜCritical Implementation Advice
1. **Backtest Relentlessly**: The Medium article provides a crucial framework. Before connecting to your live engine, you must run historical data through your logic to see how it would have performed.
2. **Start with Paper Trading**: Modify your `OrderExecutionAgent` to only log trades, not place them. Run the bot in real-time for at least two weeks to observe its behavior.
3. **Focus on Costs**: With a small capital base and high frequency for Logic A, transaction costs (brokerage, taxes, slippage) can destroy a profitable strategy. Factor these into your backtests.
4. **The Greeks Are Crucial**: Your access to real-time Delta, Theta, and Gamma is a major

advantage. Use them not just for entry, but to size positions and manage risk dynamically.

I hope this provides a robust framework for building your trading bot. Remember, this is a complex system, and these strategies are for educational purposes.

To help you refine these logics further, would you like me to elaborate on any of the following?

1. Detailed Python pseudocode for the `Signal Agent` decision tree.
2. Specific calculations for dynamic position sizing based on your ^{1}10,000 capital and Delta.
3. How to structure the trade log and analyze performance metrics post-session.