# TTW-VolumeFlow — Work Description & Project Brief

## 1) Project Overview

**TTW-VolumeFlow (NT8 Indicator)** evaluates whether current **Market (Aggressor) Volume** exceeds the **previous X bars’ volume** by a configurable multiplier.  
 To reduce false positives from extreme prints, an **ATR-based filter** dampens “Volume Spikes”. An **ATR Trailing Stop** is used to suppress opposite-side signals and preserve trend alignment.

**Goal:** Optimize the existing NinjaTrader 8 indicator (TTW-VolumeFlow.cs), enrich it functionally, and deliver a robust, user-friendly UI—without compromising runtime performance or output correctness.

## 2) Core Logic (current intent)

* **Volume Flow Condition:**  
   Current bar’s aggressor volume > *(Volume Multiplier)* × (reference volume from last *VolPeriod* bars).
  + Inputs: VolPeriod, VolumeMultiplier (and any required lookback X).
* **ATR Filter (Spike Prevention):**  
   Use ATRPeriod to suppress indications caused by extreme, low-quality spikes that often revert.
* **ATR Trailing Stop (Trend Preservation):**  
   Active stop prevents opposite-side triggers during ongoing trends.

## 3) Scope of Work

### A) C# Code Optimization

* Refactor OnStateChange/OnBarUpdate for clean state handling.
* Remove per-tick allocations; cache references; avoid boxing.
* Efficient loops & guards; minimize redundant computations.
* Correct handling for Calculate modes (OnEachTick, OnPriceChange, OnBarClose).
* Defensive coding for historical vs real-time, Tick Replay, and data alignment.

### B) Functional Enrichment

Please propose sensible additions; at minimum consider:

* Configurable **reference volume model**:
  + Highest volume of last VolPeriod bars, or
  + Moving average/EMA of volume, or
  + Robust average (trimmed mean/median) to resist outliers.
* **Directionality option** (buy/sell aggressor separation if feasible) with fallback to total volume when bid/ask data isn’t available.
* **Signal debounce** (minimum bar distance or cool-down after a trigger).
* **Output normalization** (optional) for easier cross-instrument comparison.

### C) Performance Optimization

* Target: measurable reduction in CPU time and GC pressure vs current build.
* Provide before/after notes and a simple reproduction method (same data set, same settings).
* No runtime exceptions in Logs under normal usage.

### D) UI/UX Optimization (NT8)

* **Labels & Markers:** dropdown to choose symbol/marker type; control **symbol spacing** (padding), **color**, **font family**, **font size**.
* **Panel & Plots:** clear, readable defaults; toggle for **ATR Stops** (show/hide).
* **Parameters Grouping:** tidy property grid groups (Core Logic / ATR Filter / Trailing Stop / Volume Spike Control / Alerts / UI).
* **Tooltips & XML Comments** for all public inputs.

### E) Alerts (Optional but Preferred)

* Configurable **alerts** with: enable/disable, conditions, and **customizable WAV** file path/selector.
* Basic alert throttling to avoid spam.

### F) Add-Ons (Explicit)

1. **Volume Spike Detection** to prevent false positives:
   * Option 1: threshold as a **multiplier** of the rolling baseline (VolPeriod).
   * Option 2: **Z-score** vs rolling mean & std of volume (robust to regime shifts).
   * If spike → suppress signal (and/or require confirmation).
2. **Mark Volume Spikes** visually with a **Vol Spike Signal** (plot/marker/label) and optional alert.

## 4) Deliverables

1. Updated **TTW-VolumeFlow.cs** (NT8) with:
   * Clean, documented code (XML comments) and grouped parameters.
   * Optimized performance (before/after note with simple steps to replicate).
2. **Short Technical Note** (markdown or PDF):
   * What changed + why, usage notes, edge cases, and recommended defaults.
3. **Exported Indicator Package** for easy install.
4. (If alerts enabled) **Alert usage instructions** and test steps.
5. Visuals: one or two screenshots showing labels/markers, ATR stops toggle, and spike markers.

## 5) Acceptance Criteria

* Compiles and runs on current **NinjaTrader 8** stable release.
* No runtime exceptions/warnings in the Log during standard usage.
* **Output parity** with legacy logic unless a change is explicitly agreed.
* **Performance improvement** is observable (less CPU / allocations) under identical test conditions.
* UI is tidy: readable defaults, grouped parameters, working label/marker controls.
* ATR stop toggle and spike markers behave as configured.
* Optional alerts fire as configured and can be muted.

## 6) Milestones (suggested)

* **M1 – Audit & Plan (~0.5–1 day):** Review code, confirm final param set, agree on perf targets.
* **M2 – Core Optimization:** Refactor hot paths, fix state handling, ensure parity.
* **M3 – Functional Enrichment:** Add spike logic, baseline options, debounce.
* **M4 – UI/UX + Alerts:** Labels/markers controls, ATR stop toggle, optional alerts.
* **M5 – Test & Handover:** Final polish, docs, package export, acceptance pass.

*(Hourly engagement to start; we can convert to fixed-price per milestone once the audit is done.)*

## 7) Environment & Workflow

* **Tools:** NinjaTrader 8, Visual Studio (attach for debugging), Git (branch/PR).
* **Data:** We’ll provide settings & data scenarios (historical + real-time sim) to benchmark and validate.
* **Communication:** Async updates with brief progress notes; quick turnarounds on questions.

## 8) IP & Confidentiality

* Work under our **NDA**.
* All deliverables are **work-for-hire**; exclusive IP of TTW.
* No copying/distribution without our written permission.

## 9) Open Questions for You

1. Any preference for the spike-detection method (Multiplier vs Z-Score)?
2. Can you maintain output parity for the base logic while adding new options as **disabled by default**?
3. Any UI control you’d recommend beyond labels/markers (e.g., presets)?
4. Availability and hourly rate; expected bandwidth this week.