Trading View and Dhan – Trading Management

1. I have this idea for developing a ordering system that is managed from Trading view and executed using DhanAPI. For this I have designed 3 Modules.
   1. Trading view Module – New Indicator called “PValue Pulse”
   2. Controlled Order Execution Module in Python
   3. Simple Web interface where I track my orders- That too using Python.
2. “Trading View Indicator – Pvalue Pulse” Sends following information to “Controlled Order Execution Module in Python” through web hook. “Controlled Order Execution Module” does validation, adjusts the order qty and makes several decisions and modifies the order paraments and places the order to DhanAPI. Simple Web interface fetches the order info and displays for live tracking.
3. Requirement and Design for Trading view Module – New Indicator called “PValue Pulse” using pinsescript, alerts and webhooks
   1. I want to set up multiple horizontal lines through indicator which computes that is tied with trading view alert system and then triggers the webhook. configurable paraments for that indicator as given below.
      1. Status = Options [“Live”/”Suspended”]
      2. Capital
      3. Lot Size -> default 75
      4. Capital Allocation % -> Drop down [ 25%, 50%, 75% and 100%]
      5. Price at which line to be drawn, lets call that "Initiate Trade line"
         1. This "Initiate Trade line" will act as a threshold for the price to cross, once this price is crossed, This price acts as a Limit Price. [ Draw a fixed Horizontal line at this price ]
      6. Take Profit % : % At which I want to sell all my position in profit. Default is 5%. [ Draw a fixed Horizontal line at this price ]
      7. Take Profit Trigger % : % At which I want to place my Sell order at "Take Profit %" price. Default is 3% [ Draw a fixed Horizontal line at this price ]
      8. Take Loss % : % At which I want to sell all my position in loss. Default is 15% [ Draw a fixed Horizontal line at this price ].
      9. Take Loss Trigger % : % At which I want to place my Sell order at "Take Profit %" price. Default is 12% [ Draw a fixed Horizontal line at this price ]
      10. Order Slicing Value : 1800 Default
   2. Functionality
      1. “Initiate Trade line” should be visible with Buy order webhooked alert bound to it. All other horizontal lines should be invisible.
      2. Need a table that shows following info
         1. Capital
         2. % Allocation : value from “Capital Allocation %”
         3. Lot Size
         4. Order Slicing value
         5. Total Quantity : rounddown((Capital \* % Allocation)/( Lot Size \* Price),0)\* Lot Size
         6. Number of Slices : rounddown( Quantity/ Order Slicing value,0)
         7. Non-Slice Quantity : Total Quantity – (Number of Slices\* Order Slicing value)
      3. All the below should happen only When Status = Live is Selected,
         1. Once Price moved above “Initiate Trade line”,
            1. We need to trigger buy alert webhooked to “Controlled Order Execution Module in Python” in my local machine with following order info.

Symbol name = {{ticker}} format example

NIFTY250930P25800

NIFTY : Index name

250930 :Expiry date [ YYMMDD]

P : PUT

25800 : Strike

Limit price = price of Initiate Trade line

Capital % : value from “Capital Allocation % “ dropdown

Lot Size

Order Slicing Value

ALERTNAME = “NEW BUY ORDER”

* + - * 1. Enable Take Profit Trigger % and Take Loss Trigger %. [Make it Visible]
      1. Once Price moves above Take Profit Trigger %
         1. We need to trigger sell alert webhooked to “Controlled Order Execution Module in Python” in my local machine with following order info.

Symbol name= {{ticker}}

Limit Price = Take Profit %

ALERTNAME = “PROFT BOOKING SELL”

Lot Size

Order Slicing Value

* + - * 1. Enable “Take Profit %” Horizontal line [ Make it Visible] and Make “Take Loss %” invisible if it is visible.
      1. Once Price moves below Take Loss Trigger %
         1. We need to trigger sell alert webhooked to “Controlled Order Execution Module in Python” in my local machine with following order info.

Symbol name = {{ticker}}

Limit Price = Take Loss %

ALERTNAME = “LOSS BOOKING SELL”

Lot Size

Order Slicing Value

* + - * 1. Enable “Take Loss %” Horizontal line [ Make it Visible] and Make “Take Profit %” invisible if it is visible.
      1. Once Price Moves above “Take Profit %”, Hide all Horizontal lines except “Initiate Trade line” and update Status = “Suspended”
      2. Once Price Moves below “Take Loss %”, Hide all Horizontal lines except “Initiate Trade line” and update Status = “Suspended”

1. Requirements for “Controlled Order Execution Module in Python”
   1. For now, lets build only following functionality
   2. Build a listener service and able to listen in localhost with some custom port.
   3. Print the alerts received from trading view as and when those alerts are activated.

Once we reach completing this stage I’ll provide next requirements for “Controlled Order Execution Module in Python” and “Simple Web interface”

Give the the solution in small steps so that I could learn by doing.

1. Requirements for “Controlled Order Execution Module in Python”
   1. This should be a Service that should be running on local machine, that can be started or stopped from “Simple Web interface” module.
   2. Once this service is started, this will keep listening to the Trading View Webhook signal and Execute the functions accordingly.
   3. Instrument list preparation sub-module
      1. There is a instrument list <https://api.dhan.co/v2/instrument/NSE_FNO> for FNO, whenever, the service is started, we should have separate module that downloads the csv from the above url to a pandas data frame (“df\_instrument\_key\_lookup”), keeps only those records whose “INSTRUMENT” value is in( “OPTIDX” and “FUTIDX”) and UNDERLYING\_SYMBOL in (“NIFTY”,”BANKNIFTY”). Keeps only following columns, EXCH\_ID, SECURITY\_ID, INSTRUMENT, UNDERLYING\_SYMBOL & SYMBOL\_NAME.
      2. Create a new column in the dataframe “df\_instrument\_key\_lookup” where we will convert and store the value in SYMBOL\_NAME to match the {{ticker}} format that is received from trading view.
         1. SYMBOIL\_NAME format : NIFTY-Sep2025-25800-PE
         2. {{ticker}} format : NIFTY250930P25800 – This is how we are expecting in ne column, lets call that column name as “TV\_SYMBOL\_NAME”
      3. We will use this TV\_SYMBOL\_NAME to lookup corresponding “SECURITY\_ID” that is needed for placing orders to Dhan.
   4. When Listener picks ALERTNAME = “NEW BUY ORDER”, following should happen
      1. Get the available capital from Dhan API using following