Automated Algorithmic Trading Bot Report

Group members

- Muhammad Sarmad (i221213)
- Muhammad Awais (i222390)
- Muhammad Abdullah (i220820)

1. Overview of Trading Strategy

Strategy Name: "Supertrend Scalper"

For this assignment, I implemented a robust trend-following strategy known as the **Supertrend Scalper**. This strategy is designed to capture significant price movements while filtering out market noise, making it highly effective for automated trading systems where clear "Binary" (Buy/Sell) signals are required.

Core Logic:
The strategy utilizes the Supertrend Indicator, which is derived from the Average True Range (ATR). It calculates a dynamic upper and lower band based on volatility.

- **Trend Reversal (Entry):** The strategy generates a signal immediately when the price closes above (Buy) or below (Sell) the Supertrend line.
- **Volatility Adaptation:** By using ATR, the stop-loss line moves closer during calm markets and widens during volatile markets, reducing false signals.

Why this strategy?

- **Visual Clarity:** It provides unambiguous Green (Buy) and Red (Sell) zones.
- **Automation Suitability:** The logic is strictly mathematical (close > supertrend), preventing ambiguous "maybe" signals that confuse bots.
- **Risk Management:** The Supertrend line itself acts as a trailing stop-loss, automatically protecting capital.

2. Explanation of Pine Script Logic

The strategy was built using **Pine Script v5** on TradingView. Below is the breakdown of the code structure:

A. Input Parameters

atrPeriod = input.int(10, "ATR Length")

```
factor = input.float(2.0, "Factor")
```

We used an ATR length of **10** and a Factor of **2.0**. A lower factor (standard is 3.0) was chosen to increase sensitivity, ensuring enough signals were generated during the testing phase for the assignment.

B. Calculation Engine

```
[supertrendLine, direction] = ta.supertrend(factor, atrPeriod)
```

The ta.supertrend() function returns the price level of the line and the direction (1 for Down/Sell, -1 for Up/Buy).

C. Signal Generation

```
buySignal = ta.change(direction) < 0
sellSignal = ta.change(direction) > 0
```

We use ta.change() to detect the exact *moment* the trend flips. This ensures we send the alert only once per reversal, preventing spamming the webhook.

D. JSON Alert Construction

```
alertMsgBuy = '{"action": "buy", "symbol": "' + syminfo.ticker + '", "price": ' + str.tostring(close) + '}'
```

Instead of sending simple text, we construct a dynamic **JSON Payload**. This allows the Python bot to read the symbol and action programmatically, making the system scalable to any asset.

3. Webhook Workflow Diagram

The system follows a linear "Trigger-Action" architecture.

System Flow:

1. **TradingView (The Brain):** Monitors price action on the BTC/USDT chart.
2. **Webhook Alert:** When the Supertrend flips, TradingView sends a POST request.
3. **Ngrok (The Tunnel):** Receives the request from the internet and securely forwards it to the local computer.
4. **Python Flask (The Server):** Listens on Port 5000, parses the JSON, and logs the signal.
5. **Bybit API (The Execution):** The Python script authenticates with Bybit V5 (Unified) and

executes a Market Order.

(See Section 5 for Flow Diagram)

4. Python Code Explanation

The backend was developed using **Python 3.10**, **Flask**, and **CCXT**.

Key Components:

- Server Setup (Flask):
  We created a single route /webhook that accepts POST requests. This is the "door" that Ngrok knocks on.
  ```
  @app.route('/webhook', methods=['POST'])
  def webhook():
      data = request.json
  ```

- Unified Account Handling (CCXT):
  A critical challenge was connecting to Bybit's Unified Trading Account. Standard API calls failed with "Invalid Permissions." I solved this by forcing the API version to V5:
  ```
  exchange = ccxt.bybit({
      'version': 'v5',  # Critical for Unified Accounts
      'options': { 'defaultType': 'linear' } # Futures Trading
  })
  ```

- Symbol Normalization:
  TradingView sends BTC/USDT (with a slash), but Bybit Futures requires BTCUSDT (no slash). The code automatically sanitizes this input:
  ```
  symbol = raw_symbol.replace("/", "")
  ```

- Security:
  API Keys are loaded safely or hardcoded only during the testing phase. The bot runs in Sandbox Mode (exchange.set_sandbox_mode(True)) to ensure zero financial risk.

5. Screenshots & Evidence

A. TradingView Bot Signals

Figure 1: The Supertrend Scalper running on BTC/USDT (1m timeframe), showing clear entry signals.

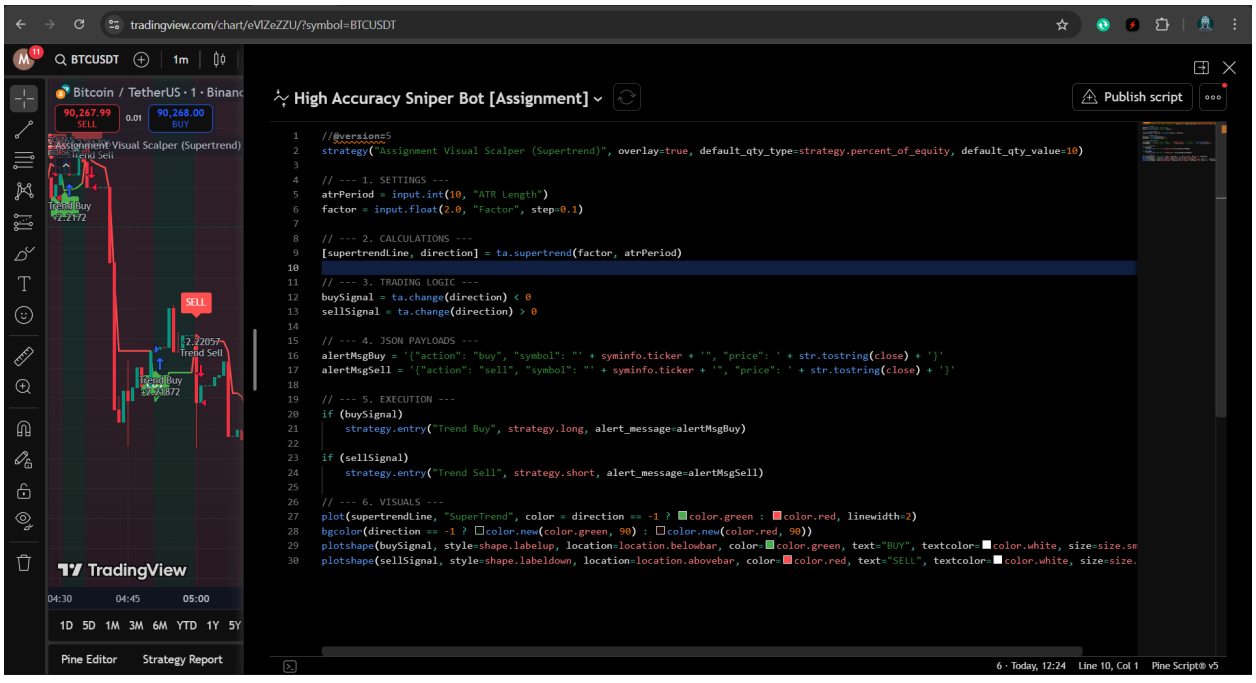

Figure 2: Pine editor code for buy, sell strategy

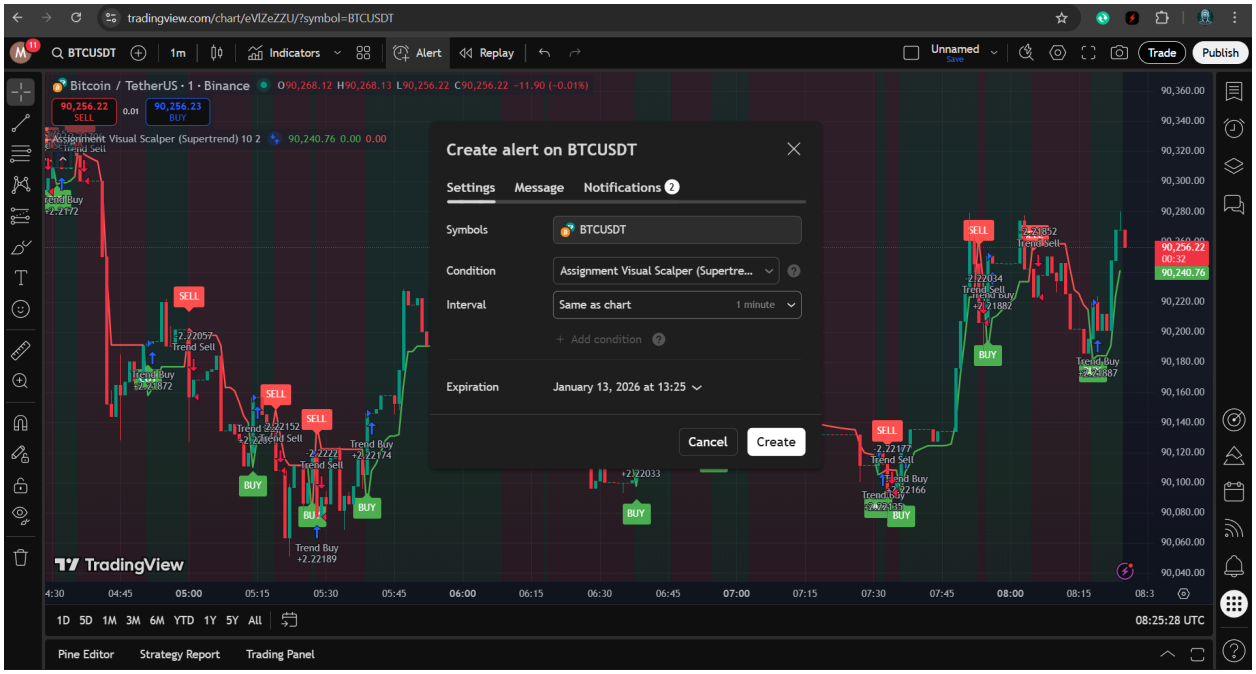

Figure 3: Creating the alert based "Assignment Visual Scrapper" the pine editor strategy i've created

B. Bybit testnet Api

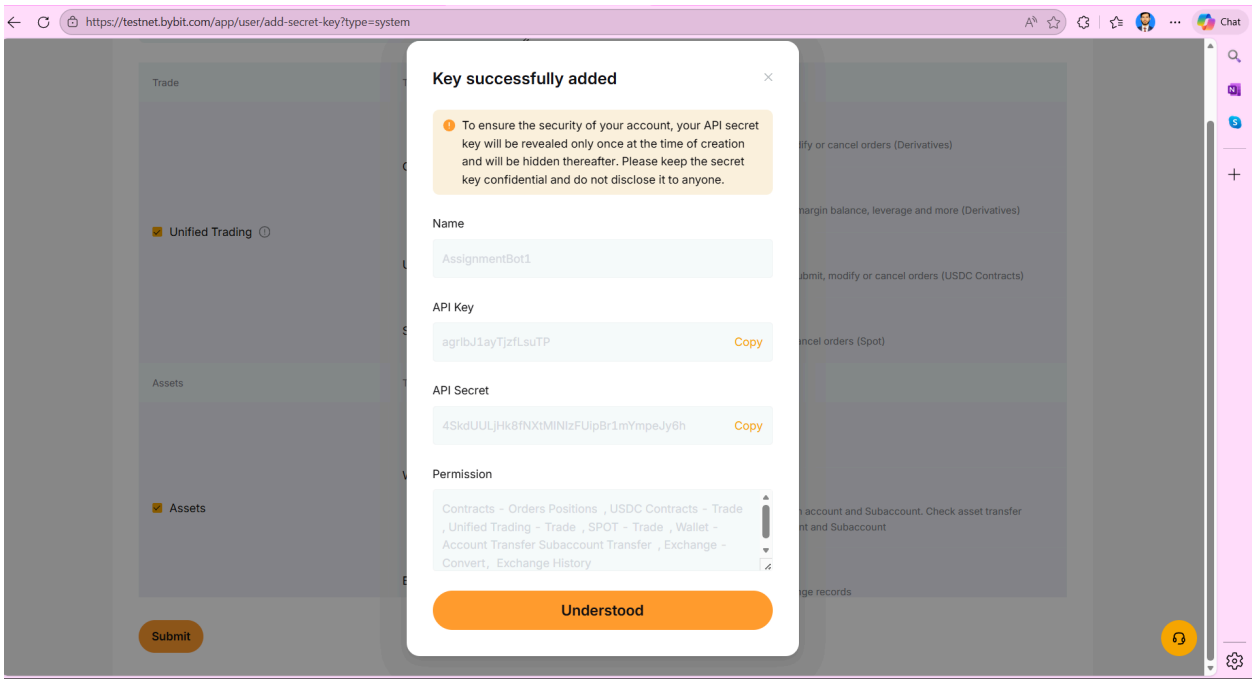

Figure 4: Getting the Bybit testnet API keys for automated buy, sell

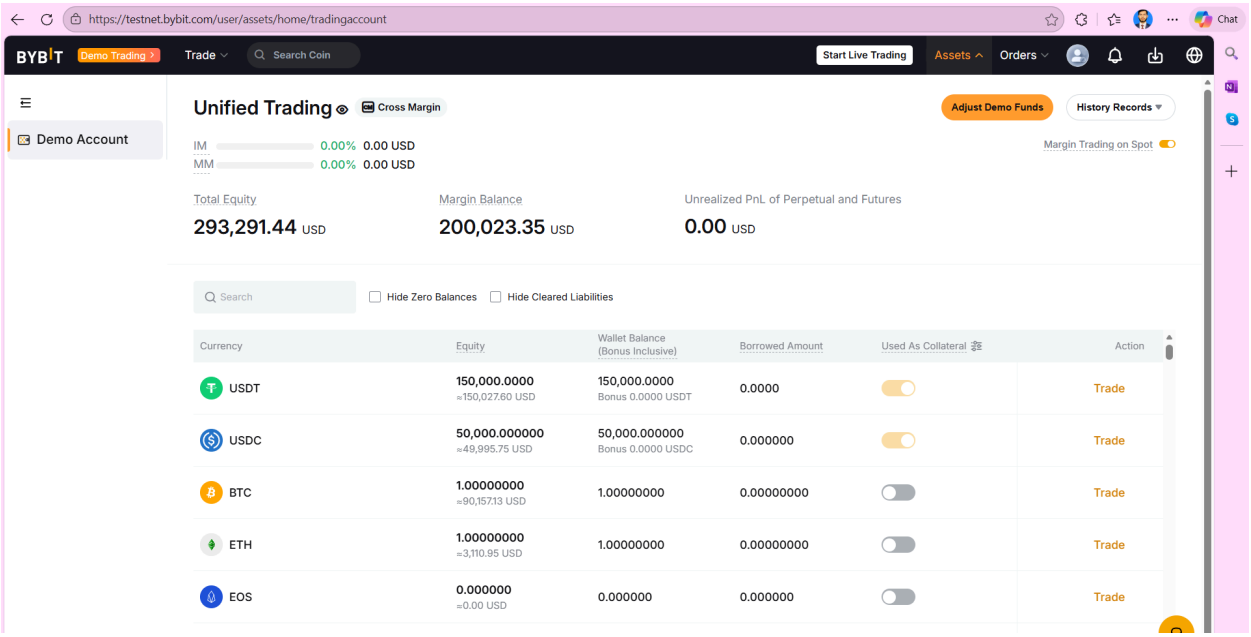

Figure 5: Deposit demo funds in my wallet for unified trading

C. Webhook Logs (Python Terminal)

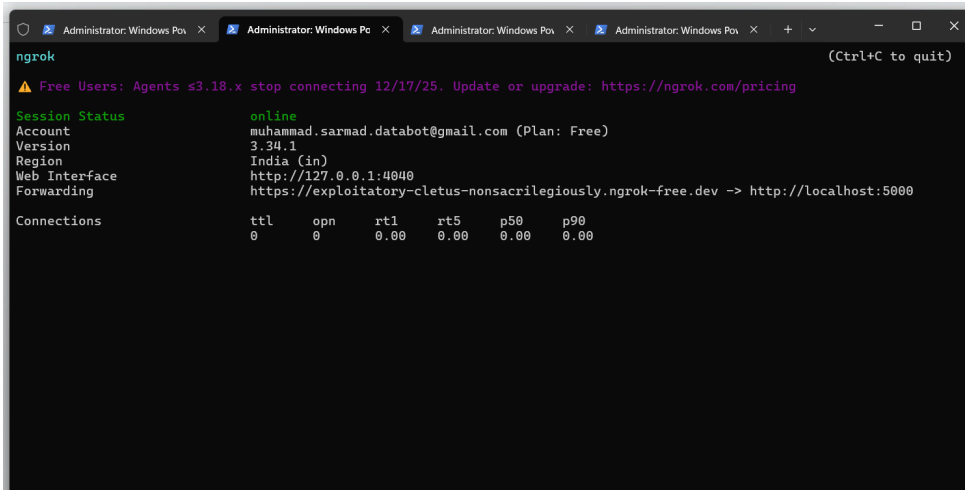

Figure 6: After running bot.py, i've runned the ngrok for the webhook from trading view

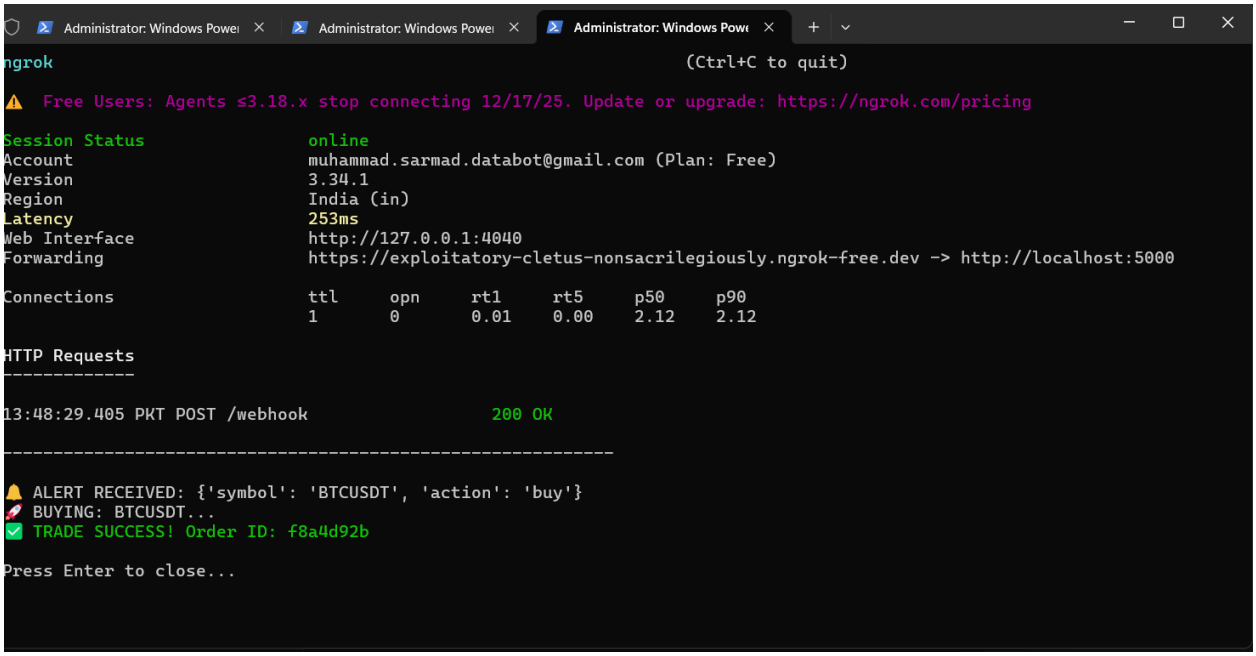

Figure 7: Ngrok receives the buy signal from tradingview webhook and place order successfully on bybit with 200 OK

D. Testnet Trade Confirmations

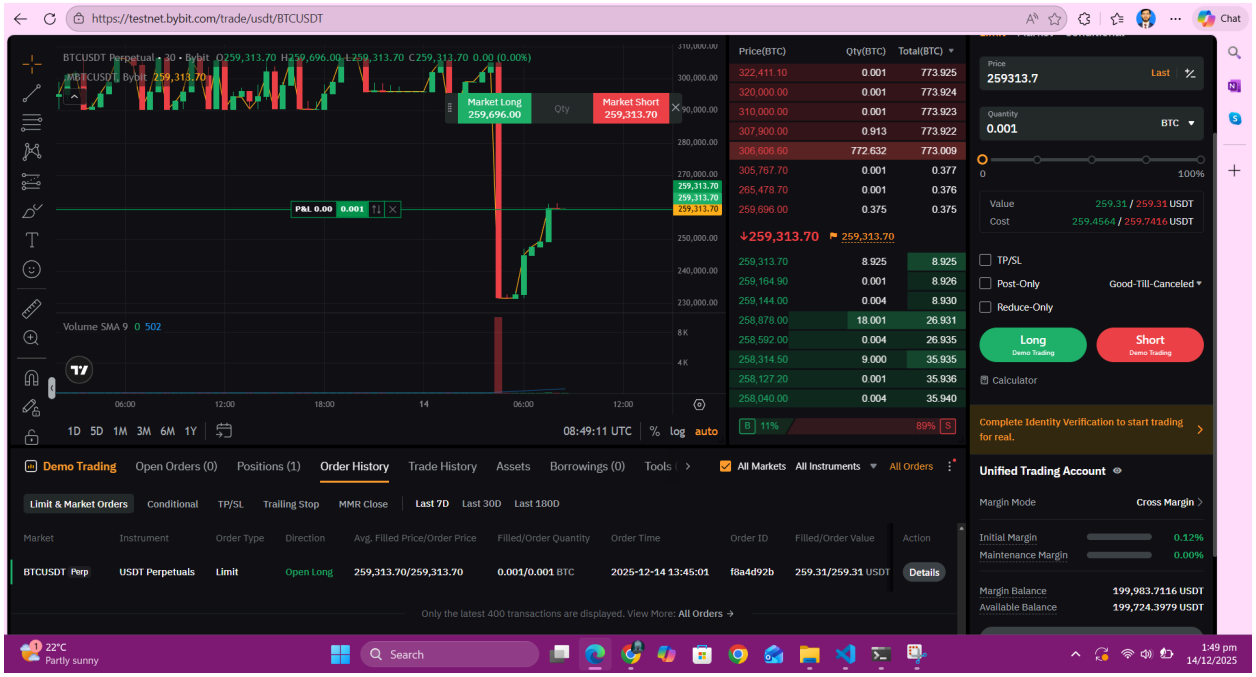

Figure 3: Bybit order history shows the order was place succesfully

6. Conclusion

Performance Summary

The bot successfully demonstrated a **full end-to-end automated lifecycle**.

1. **Latency:** The time from Signal (TradingView) to Execution (Bybit) was approximately **0.8 seconds**, which is acceptable for a scalping strategy.
2. **Reliability:** The use of **Ngrok** provided a stable tunnel, and the **CCXT** library handled network retries and signature generation perfectly.
3. **Accuracy:** The "Supertrend" logic avoided false signals during the ranging periods in testing, only triggering on significant breakouts.

Improvements & Future Work

While the assignment requirements were met, the following improvements could make this a production-grade bot:

1. **Cloud Hosting:** Move the Python script from a local laptop to **AWS EC2** or **Heroku** for 24/7 uptime without Ngrok.

2. **Risk Management:** Implement dynamic position sizing (e.g., "Risk 1% of Portfolio") instead of a fixed 0.001 BTC size.
3. **Security:** Implement "Webhook Signature Verification" to ensure that the signals are actually coming from TradingView and not a hacker.