**How to Proceed**As components increase, it get’s increasingly difficult to develop, so many indicators

We will keep it simple,   
A feature is implemented inside the feature, you can develop as many features as you want, I do not mind, but develop them individually. It will develop in **dev, uat and prod**

**Versions:**

Each feature developed, give feature version, so when it **moves to uat and prod**, we know which version of code is in which env, **no code changes in uat and dev**

Then comes the strategy: I am not worried about strategy as of now, build as many features as you can, then we will build strategies

A strategy is combination of multiple features, like RSI, support resistance, etc

Then comes final win rate and risk reward, and Kelly criteria, this we will get from the past trades, so we keep all the logs for each strategy,

Then after testing a strategy, we implement it in final code, as a strategy or working strategy, a complex strategy with parameters okay

So to make it simple, just develop as many individual features as you want.

Strategies will keep on building and maintain log and each strategy we will have backtester

Implement working tested strategy in final code as working strategy

## Features

### Implemented and validated

#### Regime detection

#### Indicators (many are implemented but can add more)

#### Plotting all indicators

#### Support resistance

#### Options data

#### Options expiry data spread

#### Options visualize strategy PL and risk plots (margin required pending)

### Implemented but need improvement or uncertain

#### Markov chain

#### News Sentiment (very little impact, segregate the quarterly earnings stock)

#### Options Greeks (implemented but not validated)

### Yet to implement

#### Volume breakout trading and corelation with the price movement

#### Pair trading

#### Options optimizer find best but put prices

#### Portfolio risk manager

## Strategies

Yet to implement

**🧠 Big Picture Improvements / Missing Components**

**1. Architecture & Code Design**

You need a **plugin-based architecture** so features and strategies can be added without touching the core.

* **Use Abstract Base Classes (ABC)** for:
  + Feature
  + Strategy
  + Backtester
* **Configuration-based execution**:
  + Run everything from a YAML/JSON config — allows flexibility without changing code.
* **Naming Convention**: Always name features with meaningful, versioned IDs (e.g., RSI\_v1, SupportRes\_v2).

**2. Backtesting Engine Enhancements**

Your backtester will become a bottleneck unless it’s highly modular and efficient.

* Multi-timeframe backtesting.
* Event-driven vs bar-by-bar simulation.
* Position sizing logic (for Kelly, Fixed %, Fixed $, Volatility-based).
* Transaction cost model — slippage, spreads, commissions.

**3. Risk Management Layer (Missing or Light)**

Most trading systems fail due to **position sizing and portfolio risk** — not strategy logic.

Implement:

* **Portfolio Risk Manager** (you mentioned this — prioritize it!)
* **Max drawdown cutoffs**.
* **Volatility filter** (avoid trades during news / high volatility).

**4. Live Trading / Paper Trading Engine**

Even if not ready yet, build with this in mind:

* Broker abstraction layer (Zerodha/Interactive Brokers/Demo).
* Order manager with retries, logs, and error tracking.
* Live vs paper mode switches.

**5. Feature Logging & Observability**

Make everything observable:

* Every feature and strategy should **log outputs** per timestamp — store in SQLite or Pandas HDF5 store.
* Include logs of:
  + Feature values
  + Strategy decisions
  + Trade entries, exits, PnL
* Build dashboards using:
  + Streamlit
  + Jupyter Dash
  + Grafana (for advanced users)

**🔍 Specific Missing Features You Might Consider**

| **Area** | **Suggested Feature** |
| --- | --- |
| Market Regimes | Volatility clustering, rolling beta |
| Trend Indicators | Donchian channels, Hull MA |
| Momentum | ROC, MOM, CCI |
| Sentiment | Twitter/Reddit scrape (low priority) |
| Correlation | Rolling correlation matrix for asset selection |
| Options | Implied volatility surface, skew |
| Portfolio | Factor models, correlation-based risk parity |

**🧪 Strategy Ideas to Implement (Once Core is Stable)**

| **Strategy** | **Brief** |
| --- | --- |
| **Breakout Strategy** | Use Donchian + Volume confirmation |
| **Mean Reversion on Pairs** | Cointegration-based (Engle-Granger, Johansen) |
| **Trend + Volatility Filter** | Trend following only in low-volatility regimes |
| **Options Short Straddle + Event Filter** | Time decay plays before expiry excluding earnings weeks |
| **Gamma Scalping** | Dynamic delta hedging + gamma capture (complex) |
| **Kelly Weighted Multi-Strategy Portfolio** | Allocate capital based on strategy edge |

**I have implemented news trading for NIFTY50, BANKNIFTY and individual stock. On top of that I can have the graph for how the news sentiment changes for a stock or index**

**I have the regime detector, which detects the regime and then I have Markov chain to predict the next day market, but the problem is that the sentiment analyzer seems lot more better predictor of the market,**

**Then I have to develop strategies, but most importantly, it is not about implementing each module, I want to implement it as accurate as it could be. So each module needs backtesting and rigorous testing**

**Combine both NIFTY and BANKNIFTY**

**16-09-2025**

**Make features folder and make branches for each feature and in each feature keep dev and uat and prod is our main code,**

**You got crest-trough points correctly, crests and troughs, now make it configurable, like parameters,  
that is base,  
then you can derive support resistance zones**

**And then you can have have fibonachi, and based on regimes, you can do range trading, or trend trading**

**Get the average trend**

**Current Implementation**

* **Regime Detection  
  Identified market regimes (Bullish, Bearish, Volatile, Neutral) using historical price and volatility behavior.**
* **Markov + Machine Learning Forecasting  
  Forecasts future Nifty50 regimes using a combination of Markov Chains and ML-based models.**
* **Technical Indicators + Price Action  
  Computed key indicators (EMA, RSI, MACD, ATR, Donchian, etc.) and price action patterns for each stock.**
* **Integrated Dataset  
  Merged stock-level data with Nifty50's regime labels and forecasts, including Nifty50 closing prices — enabling regime-aware strategy development.**
* **Strategy: Donchian Pullback (50% Midline)  
  Entry on pullback to Donchian midline (±2%) in bullish regimes, using a 2:1 risk-reward bracket.  
  ✅ Backtest completed and validated.**

**🧩 Planned Features & Next Steps**

**📊 Backtesting Framework**

* **Build a modular backtesting engine to evaluate multiple strategies.**
* **Support manual walk-forward validation using the integrated dataset.**

**📈 Stock Behavior vs Regimes**

* **Analyze each stock’s historical performance across different Nifty50 regimes.**
* **Generate metrics like:**
  + **Average return per regime**
  + **Win rate per regime**
  + **Volatility alignment**
  + **Regime sensitivity scoring**

**⚙️ Strategy Expansion**

* **Finalize and test new trading strategies.**
* **Build variants of existing strategies (e.g., Donchian + Candlestick confirmation).**
* **Develop regime-aware filters (e.g., avoid trading in “volatile” forecasts).**

**🧠 Hybrid Trading Logic**

* **Trade with more conviction when:**
  + **Current regime = bullish**
  + **Forecasted regime = bullish for next few days**
* **Reduce size or skip trades in uncertain or conflicting regime states.**
* **Build rules for confidence-weighted decision-making.**

**✅ What’s Next?**

1. **📊 Map Best Strategies to Market Regimes**
   * Choose the **best trading strategy** based on the **current detected regime**.
   * Example:
     + **Bullish Market** → Trend-following (Moving Averages, Breakouts)
     + **Bearish Market** → Short-selling, Buying Put Options
     + **High Volatility Market** → Options Trading (Straddles, Strangles)
     + **Mean Reversion Market** → Bollinger Bands, RSI Pullbacks
   * You already have a **detailed mapping** in your document.
2. **📈 Combine Predictions with Market Regime**
   * **How much should you rely on Markov Chain prediction vs. current market regime?**
   * **Hybrid Approach:**
     + If **Markov predicts the same as the current regime**, trade with **full confidence**.
     + If **Markov prediction contradicts the current regime**, reduce position size or wait for confirmation.
   * Example:
     + **Current Regime: Bullish** | **Markov Prediction: Bearish Tomorrow**  
       → Trade cautiously (Use hedges or wait for confirmation).
     + **Current Regime: Bearish** | **Markov Prediction: Bearish Tomorrow**  
       → Trade aggressively (Short sell, Buy Put Options).
3. **📌 Develop Execution Plan**
   * Set **entry & exit criteria** for each strategy.
   * Apply **risk management (stop-loss, trailing SL, profit booking).**
   * Example Execution Plan:
     + **Trend Following (Bullish Market)**
       - Buy when **price pulls back to 50EMA** and **MACD is positive**.
       - Exit at **previous resistance** or **Trailing Stop-loss**.
     + **Short Selling (Bearish Market)**
       - Short when **price rallies into resistance** and **RSI < 50**.
       - Exit when price drops **5-10% or MACD shows divergence**.
4. **📉 Backtest & Optimize Strategies**
   * Test **each strategy** on **historical data**.
   * Evaluate:
     + **Win Rate (%)**
     + **Risk-Reward Ratio**
     + **Max Drawdown**
     + **Profit Factor**
   * Optimize parameters (e.g., moving average lengths, stop-loss levels).
5. **🛠 Automate Execution (API Integration)**
   * Automate trade execution with **Zerodha Kite API / Interactive Brokers**.
   * Use a **dashboard** to monitor:
     + Live Market Regime
     + Next-Day Prediction
     + Suggested Trading Strategy
     + Real-time P&L