**📘 TÀI LIỆU DỰ ÁN: FVG RULE-BASED TRADING BOT**

**📋 MỤC LỤC**

1. [Tổng quan dự án](#1-t%E1%BB%95ng-quan-d%E1%BB%B1-%C3%A1n)
2. [Kiến trúc hệ thống](#2-ki%E1%BA%BFn-tr%C3%BAc-h%E1%BB%87-th%)
3. [Chi tiết modules](#3-chi-ti%E1%BA%BFt-modules)
4. [Luồng hoạt động](#4-lu%E1%BB%93ng-ho%E1%BA%A1t-%C4%91%E1%)
5. [Cấu hình & tham số](#5-c%E1%BA%A5u-h%C3%ACnh--tham-s%E1%BB%9)
6. [Roadmap phát triển](#6-roadmap-ph%C3%A1t-tri%E1%BB%83n)
7. [Setup & deployment](#7-setup--deployment)
8. [Testing & validation](#8-testing--validation)
9. [Troubleshooting](#9-troubleshooting)

**1. TỔNG QUAN DỰ ÁN**

**1.1. Mục tiêu**

Xây dựng bot trading **rule-based** với logic xác suất:

* **Không dùng AI/ML để dự đoán giá** (giai đoạn đầu)
* **Dựa trên xác suất thống kê** của chuỗi thua liên tiếp
* **Logic cốt lõi**: "Sau N lệnh thua liên tiếp, xác suất thắng tăng → trade thật"

**1.2. Ý tưởng chiến lược**

┌─────────────────────────────────────────────────────────┐

│ VIRTUAL MODE │

│ • Rule-based tạo tín hiệu (FVG + indicators) │

│ • Log kết quả vào file (không trade thật) │

│ • Đếm chuỗi thua liên tiếp │

│ • Khi chuỗi thua ≥ N → chuyển REAL MODE │

└─────────────────────────────────────────────────────────┘

↓

┌─────────────────────────────────────────────────────────┐

│ REAL MODE │

│ • Tín hiệu kế tiếp → mở lệnh thật │

│ • Dùng martingale (lot × 1.3^loss\_streak) │

│ • Khi WIN → reset về VIRTUAL MODE │

│ • Khi LOSS → tiếp tục tăng lot và trade │

└─────────────────────────────────────────────────────────┘

```

### \*\*1.3. Phân biệt các giai đoạn\*\*

| Giai đoạn | Mục đích | Tool | Kết nối MT5 |

|-----------|----------|------|-------------|

| \*\*Phase 1\*\* | Test chiến lược trên Python | Python backtest | ❌ Không (dùng historical data) |

| \*\*Phase 2\*\* | Live trading với tín hiệu | Python → MQL5 | ✅ Có (ZeroMQ/Socket) |

| \*\*Phase 3\*\* | Training AI với data | Python ML | ❌ Không (dùng log data) |

| \*\*Phase 4\*\* | Deploy full AI sang MQL5 | MQL5 | ✅ Có (AI nhúng trong EA) |

\*\*📍 HIỆN TẠI: Phase 1 - Python Backtest\*\*

---

## \*\*2. KIẾN TRÚC HỆ THỐNG\*\*

### \*\*2.1. Cấu trúc thư mục\*\*

```

trading\_bot/

│

├── 📄 config.py # Cấu hình toàn bộ hệ thống

├── 📄 main.py # Entry point

├── 📄 requirements.txt # Python dependencies

├── 📄 README.md # Documentation

│

├── 📁 data/ # Dữ liệu OHLCV

│ ├── download\_mt5\_data.py # Script download từ MT5

│ └── EURUSD\_M15.csv # Data file (ví dụ)

│

├── 📁 core/ # Core modules

│ ├── 📁 fvg/ # ⭐ FVG MODULE (ƯU TIÊN)

│ │ ├── fvg\_model.py # FVG object definition

│ │ ├── fvg\_detector.py # Phát hiện FVG mới

│ │ ├── fvg\_manager.py # Quản lý & tracking FVG

│ │ └── fvg\_visualizer.py # Vẽ chart FVG

│ │

│ ├── 📁 indicators/ # Technical indicators

│ │ ├── volume.py # OBV, CMF

│ │ ├── momentum.py # MACD, RSI (divergence)

│ │ ├── volatility.py # ATR, Bollinger Bands

│ │ └── trend.py # ADX

│ │

│ ├── 📁 strategy/ # Strategy logic

│ │ ├── confluence\_scorer.py # Tính điểm confluence

│ │ └── signal\_generator.py # Generate BUY/SELL

│ │

│ ├── 📁 backtest/ # Backtest engine

│ │ ├── virtual\_engine.py # Virtual trades

│ │ ├── real\_engine.py # Real trades (martingale)

│ │ ├── streak\_tracker.py # Đếm chuỗi thua

│ │ └── backtester.py # Main orchestrator

│ │

│ └── 📁 utils/ # Utilities

│ ├── data\_loader.py # Load & preprocess

│ ├── metrics.py # Performance metrics

│ └── logger.py # Custom logging

│

├── 📁 strategies/ # Specific strategies

│ └── fvg\_confluence\_strategy.py # Chiến lược chính

│

├── 📁 tests/ # Unit tests

│ ├── test\_fvg\_detector.py

│ ├── test\_fvg\_manager.py

│ └── test\_backtest.py

│

├── 📁 logs/ # Output results

│ ├── virtual\_trades.csv # Log lệnh ảo

│ ├── real\_trades.csv # Log lệnh thật

│ ├── fvg\_history.csv # Log FVG tracking

│ ├── summary.json # Metrics tổng kết

│ └── charts/ # Visualizations

│ └── EURUSD\_M15\_fvg.html

│

└── 📁 notebooks/ # Jupyter notebooks

├── fvg\_analysis.ipynb # Phân tích FVG

└── backtest\_results.ipynb # Phân tích kết quả

**2.2. Tech Stack**

| **Component** | **Technology** | **Purpose** |
| --- | --- | --- |
| **Core Language** | Python 3.10+ | Backtest & logic |
| **Data Source** | MetaTrader5 API | Historical OHLCV |
| **Indicators** | pandas-ta, ta-lib | Technical analysis |
| **Visualization** | Plotly, matplotlib | Charts & graphs |
| **Testing** | pytest | Unit tests |
| **Version Control** | Git + GitHub | Code management |
| **Future: Live Trading** | ZeroMQ / Socket | Python ↔ MQL5 |
| **Future: AI** | TensorFlow / PyTorch | ML models |

**3. CHI TIẾT MODULES**

**3.1. FVG MODULE ⭐ [ƯU TIÊN CAO NHẤT]**

**A. fvg\_model.py ✅ [HOÀN THÀNH]**

**Chức năng:**

* Định nghĩa FVG object với đầy đủ thuộc tính
* Methods để check FVG bị chạm, tính tuổi, khoảng cách

**Class chính:**

python

@dataclass

class FVG:

fvg\_id: str

fvg\_type: Literal['BULLISH', 'BEARISH']

created\_index: int

created\_timestamp: pd.Timestamp

top: float

bottom: float

middle: float *# auto-calculated*

gap\_size: float *# auto-calculated*

is\_active: bool = True

is\_touched: bool = False

strength: float = 0.0 *# gap\_size / ATR*

*# Methods:*

check\_touched() *# Check nến chạm FVG*

is\_valid\_target() *# FVG có phải target hợp lệ?*

get\_age\_in\_days() *# Tuổi FVG (90 ngày lookback)*

get\_distance\_to\_price() *# Khoảng cách tới giá*

to\_dict() *# Export to dict*

**Quy tắc quan trọng:**

* ⚠️ **FVG chạm = mất hiệu lực ngay lập tức** (không có lấp 50%)
* ⚠️ **Lookback 90 ngày** (calendar days, không phải 90 nến)
* ⚠️ **Bullish FVG chạm khi**: candle\_low <= FVG.top
* ⚠️ **Bearish FVG chạm khi**: candle\_high >= FVG.bottom

**B. fvg\_detector.py ✅ [HOÀN THÀNH]**

**Chức năng:**

* Scan 3 nến liên tiếp để tìm gap
* Validate gap (lọc noise)
* Tạo FVG object

**Class chính:**

python

class FVGDetector:

def \_\_init\_\_(self, min\_gap\_atr\_ratio=0.3, min\_gap\_pips=None):

*# min\_gap\_atr\_ratio: Gap phải >= ATR × 0.3*

*# min\_gap\_pips: Gap phải >= X pips (optional)*

def detect\_fvg\_at\_index(data, index, atr) -> FVG:

*# Phát hiện FVG tại 1 index cụ thể*

*# Check 3 nến: [i-2, i-1, i]*

def detect\_all\_fvgs(data, atr\_series) -> List[FVG]:

*# Scan toàn bộ data*

**Logic phát hiện:**

python

*# BULLISH FVG: High[i-2] < Low[i]*

if candle\_i2['high'] < candle\_i['low']:

gap\_size = candle\_i['low'] - candle\_i2['high']

if gap\_size >= ATR × 0.3:

create\_bullish\_fvg()

*# BEARISH FVG: Low[i-2] > High[i]*

if candle\_i2['low'] > candle\_i['high']:

gap\_size = candle\_i2['low'] - candle\_i['high']

if gap\_size >= ATR × 0.3:

create\_bearish\_fvg()

**C. fvg\_manager.py ⏳ [ĐANG PHÁT TRIỂN]**

**Chức năng:**

* Quản lý tất cả FVG theo thời gian thực
* Tracking trạng thái (active/touched)
* Xác định market structure (bias)
* Export FVG history

**Class chính:**

python

class FVGManager:

def \_\_init\_\_(self, lookback\_days=90):

self.active\_bullish\_fvgs = []

self.active\_bearish\_fvgs = []

self.all\_fvgs\_history = []

def update(data, current\_index, atr):

*# 1. Detect FVG mới*

*# 2. Update trạng thái FVG cũ (check touched)*

*# 3. Remove FVG > 90 ngày*

*# 4. Remove FVG đã touched*

def get\_market\_structure(current\_price) -> dict:

*# Phân tích FVG structure*

*# Returns:*

{

'bias': 'BULLISH\_BIAS' | 'BEARISH\_BIAS' | 'BOTH\_FVG' | 'NO\_FVG',

'bullish\_fvgs\_below': [...],

'bearish\_fvgs\_above': [...],

'nearest\_bullish\_target': FVG,

'nearest\_bearish\_target': FVG

}

**Logic market structure:**

| **Tình huống** | **FVG dưới** | **FVG trên** | **Bias** | **Action** |
| --- | --- | --- | --- | --- |
| Case 1 | ✅ Có | ❌ Không | BULLISH\_BIAS | Chỉ tìm BUY |
| Case 2 | ❌ Không | ✅ Có | BEARISH\_BIAS | Chỉ tìm SELL |
| Case 3 | ✅ Có | ✅ Có | BOTH\_FVG | Theo indicators |
| Case 4 | ❌ Không | ❌ Không | NO\_FVG | ❌ NO TRADE |

**⚠️ QUY TẮC VÀNG:**

python

if bias == 'BULLISH\_BIAS':

if indicators\_signal == 'BUY':

✅ TRADE (có target FVG dưới)

else:

❌ NO TRADE

elif bias == 'BEARISH\_BIAS':

if indicators\_signal == 'SELL':

✅ TRADE (có target FVG trên)

else:

❌ NO TRADE

elif bias == 'BOTH\_FVG':

if indicators\_signal == 'BUY':

✅ TRADE BUY (target FVG dưới)

elif indicators\_signal == 'SELL':

✅ TRADE SELL (target FVG trên)

elif bias == 'NO\_FVG':

❌ NO TRADE (không có target)

**D. fvg\_visualizer.py ⏳ [ĐANG PHÁT TRIỂN]**

**Chức năng:**

* Vẽ chart OHLC với FVG zones
* Highlight FVG active vs touched
* Visualize signals & trades

**Output:**

python

*# HTML interactive chart (Plotly)*

*# - Candlestick chart*

*# - FVG zones (rectangles)*

*# - Green: Bullish FVG active*

*# - Red: Bearish FVG active*

*# - Gray: FVG touched*

*# - Entry/exit markers*

*# - Annotations (confluence score, etc.)*

**3.2. INDICATORS MODULE ⏳ [TODO]**

**A. volume.py**

**Indicators cần implement:**

1. **OBV (On-Balance Volume)**

python

def calculate\_obv(data) -> pd.Series:

*# Logic:*

*# - Nến xanh: OBV += volume*

*# - Nến đỏ: OBV -= volume*

def detect\_obv\_divergence(data, obv, window=14) -> dict:

*# Bullish divergence:*

*# - Price: lower low*

*# - OBV: higher low*

*# Returns: {'bullish': bool, 'bearish': bool}*

1. **CMF (Chaikin Money Flow)**

python

def calculate\_cmf(data, period=20) -> pd.Series:

*# Logic:*

*# MFM = ((Close - Low) - (High - Close)) / (High - Low)*

*# MFV = MFM × Volume*

*# CMF = Sum(MFV, period) / Sum(Volume, period)*

*# Thresholds:*

*# CMF > +0.05: Strong buying pressure*

*# CMF < -0.05: Strong selling pressure*

**Confluence scoring:**

python

*# BUY score*

if obv\_bullish\_divergence: +2

if obv\_leading\_breakout: +1

if cmf > 0.05: +2

*# SELL score*

if obv\_bearish\_divergence: +2

if obv\_leading\_breakdown: +1

if cmf < -0.05: +2

**B. momentum.py**

**Indicators cần implement:**

1. **MACD (Moving Average Convergence Divergence)**

python

def calculate\_macd(data, fast=12, slow=26, signal=9) -> dict:

*# Returns: {*

*# 'macd': Series,*

*# 'signal': Series,*

*# 'histogram': Series*

*# }*

def detect\_macd\_divergence(data, macd, window=14) -> dict:

*# Bullish: Price ↓, MACD ↑*

*# Bearish: Price ↑, MACD ↓*

1. **RSI (Relative Strength Index)**

python

def calculate\_rsi(data, period=14) -> pd.Series:

*# Standard RSI calculation*

def detect\_rsi\_divergence(data, rsi, window=14) -> dict:

*# Similar to MACD divergence*

**Confluence scoring:**

python

*# BUY score*

if macd\_bullish\_divergence: +3

if macd\_cross\_signal\_up: +1

if histogram\_increasing: +1

if rsi < 30 and rsi\_bullish\_div: +2

if rsi < 20: +1 *# Extreme oversold*

*# SELL score*

if macd\_bearish\_divergence: +3

if macd\_cross\_signal\_down: +1

if histogram\_decreasing: +1

if rsi > 70 and rsi\_bearish\_div: +2

if rsi > 80: +1 *# Extreme overbought*

**C. volatility.py**

**Indicators cần implement:**

1. **ATR (Average True Range)**

python

def calculate\_atr(data, period=14) -> pd.Series:

*# True Range = max(*

*# high - low,*

*# abs(high - close\_prev),*

*# abs(low - close\_prev)*

*# )*

*# ATR = SMA(TR, period)*

*# Usage:*

*# - Dynamic SL/TP: SL = entry ± (1.5 × ATR)*

*# - Environment filter: ATR > threshold*

1. **Bollinger Bands**

python

def calculate\_bollinger\_bands(data, period=20, std=2) -> dict:

*# Returns: {*

*# 'upper': Series,*

*# 'middle': Series (SMA),*

*# 'lower': Series,*

*# 'percent\_b': Series, # (%B indicator)*

*# 'bandwidth': Series*

*# }*

*# %B = (Price - Lower) / (Upper - Lower)*

*# %B > 1: Price above upper band*

*# %B < 0: Price below lower band*

**Confluence scoring:**

python

*# BUY score*

if bb\_percent\_b < 0: +2 *# Price below lower band*

if bb\_bandwidth > threshold: +1 *# Market moving*

*# SELL score*

if bb\_percent\_b > 1: +2 *# Price above upper band*

if bb\_bandwidth > threshold: +1

**D. trend.py**

**Indicator cần implement:**

**ADX (Average Directional Index)**

python

def calculate\_adx(data, period=14) -> dict:

*# Returns: {*

*# 'adx': Series,*

*# 'di\_plus': Series,*

*# 'di\_minus': Series*

*# }*

*# Interpretation:*

*# ADX > 25: Có trend (tradeable)*

*# ADX > 40: Trend rất mạnh*

*# ADX < 20: Sideway (NO TRADE)*

*# Trend direction:*

*# DI+ > DI- and ADX rising: Uptrend*

*# DI- > DI+ and ADX rising: Downtrend*

**Environment filter:**

python

*# CRITICAL: Block trade khi sideway*

if adx < 25:

return "NO\_TRADE" *# Sideway market*

**3.3. STRATEGY MODULE ⏳ [TODO]**

**A. confluence\_scorer.py**

**Chức năng:**

* Tính điểm confluence cho BUY/SELL
* Aggregate scores từ tất cả indicators

python

class ConfluenceScorer:

def calculate\_scores(self, data, indicators, fvg\_structure) -> dict:

"""

Returns: {

'buy\_score': int (0-14),

'sell\_score': int (0-14),

'buy\_breakdown': dict,

'sell\_breakdown': dict

}

"""

buy\_score = 0

sell\_score = 0

*# === VOLUME (5 điểm) ===*

if indicators['obv\_bull\_div']:

buy\_score += 2

if indicators['obv\_bear\_div']:

sell\_score += 2

if indicators['cmf'] > 0.05:

buy\_score += 2

if indicators['cmf'] < -0.05:

sell\_score += 2

*# === DIVERGENCE (7 điểm) ===*

if indicators['macd\_bull\_div']:

buy\_score += 3

if indicators['macd\_bear\_div']:

sell\_score += 3

if indicators['rsi'] < 30 and indicators['rsi\_bull\_div']:

buy\_score += 2

if indicators['rsi'] > 70 and indicators['rsi\_bear\_div']:

sell\_score += 2

*# === VOLATILITY (2 điểm) ===*

if indicators['bb\_percent\_b'] < 0:

buy\_score += 2

if indicators['bb\_percent\_b'] > 1:

sell\_score += 2

return {

'buy\_score': buy\_score,

'sell\_score': sell\_score,

'total\_possible': 14

}

**Thresholds:**

python

*# Entry requirements*

MIN\_CONFLUENCE\_SCORE = 7 *# >= 7 điểm mới trade*

HIGH\_CONFIDENCE\_SCORE = 10 *# >= 10 điểm = high confidence*

*# Position sizing*

if score >= 10:

risk\_percent = 2.0 *# High confidence*

elif score >= 7:

risk\_percent = 1.0 *# Medium confidence*

**B. signal\_generator.py**

**Chức năng:**

* Kết hợp FVG + Confluence + Environment
* Generate signal cuối cùng

python

class SignalGenerator:

def generate\_signal(self, data, fvg\_structure, indicators) -> dict:

"""

Main signal generation logic

Returns:

{

'signal': 'BUY' | 'SELL' | None,

'entry': float,

'confluence\_score': int,

'fvg\_target': FVG,

'confidence': 'HIGH' | 'MEDIUM',

'reasons': [...]

}

"""

*# === STEP 1: Environment Filter ===*

if indicators['adx'] < 25:

return None *# Sideway*

if indicators['atr'] < min\_threshold:

return None *# Low volatility*

*# === STEP 2: Calculate Confluence ===*

scores = confluence\_scorer.calculate\_scores(...)

buy\_score = scores['buy\_score']

sell\_score = scores['sell\_score']

*# === STEP 3: Determine Signal ===*

if buy\_score >= 7 and buy\_score > sell\_score:

signal\_direction = 'BUY'

score = buy\_score

elif sell\_score >= 7 and sell\_score > buy\_score:

signal\_direction = 'SELL'

score = sell\_score

else:

return None *# Not enough confluence*

*# === STEP 4: Validate with FVG ===*

is\_valid = self.\_validate\_fvg(fvg\_structure, signal\_direction)

if not is\_valid:

return None *# No FVG target*

*# === STEP 5: Build Signal ===*

return {

'signal': signal\_direction,

'entry': data['close'].iloc[-1],

'confluence\_score': score,

'confidence': 'HIGH' if score >= 10 else 'MEDIUM',

'fvg\_target': self.\_get\_fvg\_target(fvg\_structure, signal\_direction),

'timestamp': data.index[-1]

}

def \_validate\_fvg(self, fvg\_structure, signal\_direction) -> bool:

"""Validate theo logic FVG"""

bias = fvg\_structure['bias']

if signal\_direction == 'BUY':

return bias in ['BULLISH\_BIAS', 'BOTH\_FVG']

elif signal\_direction == 'SELL':

return bias in ['BEARISH\_BIAS', 'BOTH\_FVG']

return False

**3.4. BACKTEST MODULE ⏳ [TODO]**

**A. virtual\_engine.py**

**Chức năng:**

* Chạy virtual trades (không dùng vốn thật)
* Log kết quả
* Track chuỗi thua

python

class VirtualEngine:

def \_\_init\_\_(self, sl\_tp\_config):

self.trades = []

self.loss\_streak = 0

def execute\_virtual\_trade(self, signal, data, future\_data) -> dict:

"""

Simulate trade outcome

Args:

signal: Signal dict from generator

data: Current data

future\_data: Next N candles để check SL/TP

Returns:

{

'outcome': 'WIN' | 'LOSS',

'pnl\_pips': float,

'exit\_price': float,

'exit\_reason': 'TP' | 'SL',

'bars\_held': int

}

"""

entry = signal['entry']

direction = signal['signal']

*# Calculate SL/TP (FIXED PIPS)*

sl\_pips = self.sl\_tp\_config['sl\_pips']

tp\_pips = self.sl\_tp\_config['tp\_pips']

if direction == 'BUY':

sl = entry - (sl\_pips \* 0.0001)

tp = entry + (tp\_pips \* 0.0001)

else:

sl = entry + (sl\_pips \* 0.0001)

tp = entry - (tp\_pips \* 0.0001)

*# Simulate: Check each future candle*

for i, candle in future\_data.iterrows():

if direction == 'BUY':

if candle['low'] <= sl:

*# Hit SL*

return {

'outcome': 'LOSS',

'pnl\_pips': -sl\_pips,

'exit\_price': sl,

'exit\_reason': 'SL',

'bars\_held': i

}

elif candle['high'] >= tp:

*# Hit TP*

return {

'outcome': 'WIN',

'pnl\_pips': tp\_pips,

'exit\_price': tp,

'exit\_reason': 'TP',

'bars\_held': i

}

*# Similar for SELL*

*# Timeout (không hit SL/TP trong N nến)*

return {'outcome': 'TIMEOUT', ...}

def update\_streak(self, outcome):

"""Update loss streak counter"""

if outcome == 'LOSS':

self.loss\_streak += 1

else:

self.loss\_streak = 0

def should\_trigger\_real\_mode(self, trigger\_threshold) -> bool:

"""Check if should switch to REAL mode"""

return self.loss\_streak >= trigger\_threshold

**B. real\_engine.py**

**Chức năng:**

* Chạy real trades với martingale
* Tính lot size động
* Track real loss streak

python

class RealEngine:

def \_\_init\_\_(self, base\_lot, martingale\_factor, max\_steps):

self.base\_lot = base\_lot

self.martingale\_factor = martingale\_factor

self.max\_steps = max\_steps

self.real\_loss\_streak = 0

self.trades = []

def execute\_real\_trade(self, signal, data, future\_data) -> dict:

"""

Execute real trade with martingale

Lot size = base\_lot × (martingale\_factor ^ real\_loss\_streak)

"""

*# Calculate lot*

lot\_size = self.base\_lot \* (self.martingale\_factor \*\* self.real\_loss\_streak)

*# Limit max steps*

if self.real\_loss\_streak >= self.max\_steps:

lot\_size = self.base\_lot \* (self.martingale\_factor \*\* self.max\_steps)

*# Simulate trade (same as virtual)*

result = self.\_simulate\_trade(signal, data, future\_data)

*# Calculate PnL in dollars*

pnl\_dollars = result['pnl\_pips'] \* 10 \* lot\_size *# Forex: $10/pip/lot*

result['lot\_size'] = lot\_size

result['pnl\_dollars'] = pnl\_dollars

*# Update streak*

if result['outcome'] == 'WIN':

self.real\_loss\_streak = 0 *# RESET*

elif result['outcome'] == 'LOSS':

self.real\_loss\_streak += 1

self.trades.append(result)

return result

def should\_return\_virtual\_mode(self, last\_outcome) -> bool:

"""After WIN, return to VIRTUAL"""

return last\_outcome == 'WIN'

**C. streak\_tracker.py**

**Chức năng:**

* Track chuỗi thua virtual & real
* Thống kê streak patterns

python

class StreakTracker:

def \_\_init\_\_(self):

self.virtual\_streaks = []

self.real\_streaks = []

self.current\_virtual\_streak = 0

self.current\_real\_streak = 0

def update\_virtual(self, outcome):

if outcome == 'LOSS':

self.current\_virtual\_streak += 1

else:

if self.current\_virtual\_streak > 0:

self.virtual\_streaks.append(self.current\_virtual\_streak)

self.current\_virtual\_streak = 0

def get\_statistics(self) -> dict:

"""

Returns:

{

'max\_virtual\_streak': int,

'max\_real\_streak': int,

'avg\_virtual\_streak': float,

'streak\_distribution': dict

}

"""

**D. backtester.py - MAIN ORCHESTRATOR**

**Chức năng:**

* Kết nối tất cả modules
* Chạy backtest loop
* Export results

python

class Backtester:

def \_\_init\_\_(self, strategy, config):

self.strategy = strategy

self.config = config

*# Initialize components*

self.fvg\_manager = FVGManager()

self.virtual\_engine = VirtualEngine(config['sl\_tp'])

self.real\_engine = RealEngine(config['base\_lot'], ...)

self.streak\_tracker = StreakTracker()

self.mode = "VIRTUAL" *# VIRTUAL hoặc REAL*

def run(self, data) -> dict:

"""

Main backtest loop

For each candle:

1. Update FVG manager

2. Calculate indicators

3. Generate signal (if environment OK)

4. Validate with FVG

5. Execute trade (virtual/real based on mode)

6. Update streaks

7. Switch mode if needed

Returns:

{

'virtual\_trades': [...],

'real\_trades': [...],

'metrics': {...},

'fvg\_history': [...]

}

"""

results = {

'virtual\_trades': [],

'real\_trades': [],

'fvg\_history': []

}

for i in range(100, len(data)):

*# === UPDATE FVG ===*

atr = calculate\_atr(data[:i+1])

self.fvg\_manager.update(data[:i+1], i, atr)

fvg\_structure = self.fvg\_manager.get\_market\_structure(data['close'].iloc[i])

*# === CALCULATE INDICATORS ===*

indicators = self.strategy.calculate\_all\_indicators(data[:i+1])

*# === GENERATE SIGNAL ===*

signal = self.strategy.generate\_signal(data[:i+1], fvg\_structure, indicators)

if signal is None:

continue

*# === EXECUTE TRADE ===*

future\_data = data[i+1:i+51] *# Next 50 candles*

if self.mode == "VIRTUAL":

trade\_result = self.virtual\_engine.execute\_virtual\_trade(signal, data[:i+1], future\_data)

results['virtual\_trades'].append(trade\_result)

*# Update streak*

self.virtual\_engine.update\_streak(trade\_result['outcome'])

*# Check trigger*

if self.virtual\_engine.should\_trigger\_real\_mode(self.config['loss\_streak\_trigger']):

self.mode = "REAL"

print(f"🔴 SWITCH TO REAL MODE at {data.index[i]}")

elif self.mode == "REAL":

trade\_result = self.real\_engine.execute\_real\_trade(signal, data[:i+1], future\_data)

results['real\_trades'].append(trade\_result)

*# Check return to virtual*

if self.real\_engine.should\_return\_virtual\_mode(trade\_result['outcome']):

self.mode = "VIRTUAL"

self.virtual\_engine.loss\_streak = 0 *# Reset*

print(f"🟢 SWITCH TO VIRTUAL MODE at {data.index[i]}")

*# Calculate metrics*

results['metrics'] = self.\_calculate\_metrics(results)

return results

def \_calculate\_metrics(self, results) -> dict:

"""Calculate performance metrics"""

virtual\_trades = results['virtual\_trades']

real\_trades = results['real\_trades']

*# Virtual stats*

v\_wins = sum(1 for t in virtual\_trades if t['outcome'] == 'WIN')

v\_losses = sum(1 for t in virtual\_trades if t['outcome'] == 'LOSS')

v\_win\_rate = v\_wins / len(virtual\_trades) if virtual\_trades else 0

*# Real stats*

r\_wins = sum(1 for t in real\_trades if t['outcome'] == 'WIN')

r\_losses = sum(1 for t in real\_trades if t['outcome'] == 'LOSS')

r\_win\_rate = r\_wins / len(real\_trades) if real\_trades else 0

*# PnL*

total\_pnl = sum(t['pnl\_dollars'] for t in real\_trades)

*# Drawdown*

equity\_curve = self.\_build\_equity\_curve(real\_trades)

max\_dd = self.\_calculate\_max\_drawdown(equity\_curve)

return {

'virtual\_trades\_count': len(virtual\_trades),

'virtual\_win\_rate': v\_win\_rate,

'real\_trades\_count': len(real\_trades),

'real\_win\_rate': r\_win\_rate,

'total\_pnl': total\_pnl,

'max\_drawdown': max\_dd,

'sharpe\_ratio': self.\_calculate\_sharpe(equity\_curve),

'profit\_factor': self.\_calculate\_profit\_factor(real\_trades)

}

```

---

*## \*\*4. LUỒNG HOẠT ĐỘNG\*\**

*### \*\*4.1. Flow Chart Tổng Quan\*\**

```

START

↓

[1] Load Data (OHLCV from MT5 or CSV)

↓

[2] Initialize Components

• FVGManager

• Indicators calculators

• Strategy

• Virtual/Real engines

↓

[3] Loop qua từng nến (i = 100 → end)

↓

[3.1] Update FVG Manager

• Detect FVG mới

• Update trạng thái FVG cũ (check touched)

• Remove FVG > 90 ngày hoặc touched

• Get market structure (bias)

↓

[3.2] Calculate Indicators

• ATR, ADX (environment)

• OBV, CMF (volume)

• MACD, RSI (divergence)

• Bollinger Bands (volatility)

↓

[3.3] Environment Filter

• ADX >= 25? → Continue

• ATR > threshold? → Continue

• Else → Skip to next candle

↓

[3.4] Calculate Confluence Score

• Volume: OBV div, CMF

• Divergence: MACD div, RSI div

• Volatility: BB %B

• Total: 0-14 điểm

↓

[3.5] Generate Signal

• Score >= 7? → Continue

• Determine BUY/SELL

• Else → Skip

↓

[3.6] Validate with FVG

• Check FVG structure

• Apply QUY TẮC VÀNG

• Valid? → Continue

• Else → Skip

↓

[3.7] Execute Trade (based on MODE)

┌─────────────────┬─────────────────┐

│ VIRTUAL MODE │ REAL MODE │

├─────────────────┼─────────────────┤

│ • Lot = 0 │ • Lot = base × │

│ • Log only │ (1.3^streak) │

│ • Update streak │ • Update equity │

│ │ • Update streak │

└─────────────────┴─────────────────┘

↓

[3.8] Check Mode Switch

• VIRTUAL: streak >= 3? → REAL

• REAL: WIN? → VIRTUAL

↓

[4] Export Results

• virtual\_trades.csv

• real\_trades.csv

• fvg\_history.csv

• summary.json

↓

[5] Generate Visualizations

• FVG chart

• Equity curve

• Drawdown chart

↓

END

```

---

*### \*\*4.2. Decision Tree - Signal Validation\*\**

```

Signal Generated?

├─ NO → Skip to next candle

└─ YES

↓

Environment OK? (ADX >= 25, ATR > threshold)

├─ NO → Skip

└─ YES

↓

Confluence Score >= 7?

├─ NO → Skip

└─ YES

↓

FVG Structure Check:

├─ BULLISH\_BIAS (FVG dưới, không có FVG trên)

│ ├─ Signal = BUY? → ✅ TRADE

│ └─ Signal = SELL? → ❌ REJECT

│

├─ BEARISH\_BIAS (FVG trên, không có FVG dưới)

│ ├─ Signal = SELL? → ✅ TRADE

│ └─ Signal = BUY? → ❌ REJECT

│

├─ BOTH\_FVG (có cả 2 phía)

│ ├─ Signal = BUY? → ✅ TRADE (target FVG dưới)

│ └─ Signal = SELL? → ✅ TRADE (target FVG trên)

│

└─ NO\_FVG (không có FVG nào)

→ ❌ REJECT (không có target)

**5. CẤU HÌNH & THAM SỐ**

**5.1. config.py - File cấu hình chính**

python

*# config.py*

"""

Cấu hình toàn bộ hệ thống trading bot

"""

*# ============================================*

*# DATA CONFIGURATION*

*# ============================================*

DATA\_CONFIG = {

'symbol': 'EURUSD',

'timeframe': 'M15', *# M1, M5, M15, M30, H1, H4, D1*

'start\_date': '2024-01-01',

'end\_date': '2024-12-31',

'data\_source': 'MT5', *# MT5 hoặc CSV*

'csv\_path': 'data/EURUSD\_M15.csv' *# Nếu dùng CSV*

}

*# ============================================*

*# FVG CONFIGURATION*

*# ============================================*

FVG\_CONFIG = {

'lookback\_days': 90, *# FVG chỉ valid trong 90 ngày*

'min\_gap\_atr\_ratio': 0.3, *# Gap phải >= ATR × 0.3*

'min\_gap\_pips': None, *# Optional: Gap phải >= X pips*

'enable\_visualization': True

}

*# ============================================*

*# INDICATORS CONFIGURATION*

*# ============================================*

INDICATORS\_CONFIG = {

*# ATR*

'atr\_period': 14,

'min\_atr\_threshold': 0.0010, *# Tối thiểu ATR để trade*

*# ADX*

'adx\_period': 14,

'min\_adx': 25, *# ADX >= 25 mới trade*

*# Bollinger Bands*

'bb\_period': 20,

'bb\_std': 2,

*# MACD*

'macd\_fast': 12,

'macd\_slow': 26,

'macd\_signal': 9,

*# RSI*

'rsi\_period': 14,

'rsi\_oversold': 30,

'rsi\_overbought': 70,

*# OBV & CMF*

'cmf\_period': 20,

'cmf\_threshold': 0.05, *# CMF > +0.05 hoặc < -0.05*

*# Divergence detection*

'divergence\_window': 14

}

*# ============================================*

*# STRATEGY CONFIGURATION*

*# ============================================*

STRATEGY\_CONFIG = {

'min\_confluence\_score': 7, *# Tối thiểu 7/14 điểm*

'high\_confidence\_score': 10, *# >= 10 điểm = high confidence*

}

*# ============================================*

*# SL/TP STRATEGIES*

*# ============================================*

SL\_TP\_STRATEGIES = {

'conservative': {

'sl\_pips': 20,

'tp\_pips': 40,

'rr\_ratio': 2.0

},

'balanced': {

'sl\_pips': 20,

'tp\_pips': 50,

'rr\_ratio': 2.5

},

'aggressive': {

'sl\_pips': 20,

'tp\_pips': 60,

'rr\_ratio': 3.0

},

'tight': {

'sl\_pips': 15,

'tp\_pips': 30,

'rr\_ratio': 2.0

},

'wide': {

'sl\_pips': 30,

'tp\_pips': 60,

'rr\_ratio': 2.0

}

}

*# ============================================*

*# BACKTEST CONFIGURATION*

*# ============================================*

BACKTEST\_CONFIG = {

*# Virtual → Real trigger*

'loss\_streak\_trigger\_options': [2, 3, 4, 5], *# Test nhiều giá trị*

*# Martingale*

'base\_lot': 0.1,

'martingale\_factor': 1.3,

'max\_martingale\_steps': 4,

*# Risk management*

'risk\_percent': 2.0, *# % account mỗi lệnh*

'starting\_balance': 10000,

*# Trade simulation*

'max\_bars\_to\_hold': 50, *# Timeout nếu không hit SL/TP*

'slippage\_pips': 1 *# Slippage simulation*

}

*# ============================================*

*# TEST MATRIX (Multiple backtests)*

*# ============================================*

TEST\_MATRIX = {

'test\_1': {

'sl\_tp\_strategy': 'conservative',

'loss\_streak\_trigger': 3,

'description': 'Conservative SL/TP, trigger=3'

},

'test\_2': {

'sl\_tp\_strategy': 'balanced',

'loss\_streak\_trigger': 3,

'description': 'Balanced SL/TP, trigger=3'

},

'test\_3': {

'sl\_tp\_strategy': 'aggressive',

'loss\_streak\_trigger': 3,

'description': 'Aggressive SL/TP, trigger=3'

},

'test\_4': {

'sl\_tp\_strategy': 'conservative',

'loss\_streak\_trigger': 4,

'description': 'Conservative SL/TP, trigger=4'

},

*# ... thêm nhiều tests*

}

*# ============================================*

*# LOGGING & OUTPUT*

*# ============================================*

LOGGING\_CONFIG = {

'log\_level': 'INFO', *# DEBUG, INFO, WARNING, ERROR*

'log\_to\_file': True,

'log\_file': 'logs/bot.log',

*# Output files*

'virtual\_trades\_csv': 'logs/virtual\_trades.csv',

'real\_trades\_csv': 'logs/real\_trades.csv',

'fvg\_history\_csv': 'logs/fvg\_history.csv',

'summary\_json': 'logs/summary.json',

*# Charts*

'charts\_dir': 'logs/charts/',

'save\_charts': True

}

*# ============================================*

*# MT5 INTEGRATION (Phase 2)*

*# ============================================*

MT5\_CONFIG = {

'enabled': False, *# Phase 1: False, Phase 2: True*

'login': 12345678,

'password': 'your\_password',

'server': 'Broker-Server',

'timeout': 60000,

'magic\_number': 123456

}

**5.2. Các tham số quan trọng cần optimize**

| **Tham số** | **Mặc định** | **Mô tả** | **Test range** |
| --- | --- | --- | --- |
| **loss\_streak\_trigger** | 3 | Số lệnh ảo thua liên tiếp để kích hoạt real | 2, 3, 4, 5 |
| **min\_confluence\_score** | 7 | Điểm tối thiểu để trade | 5, 6, 7, 8 |
| **sl\_pips** | 20 | Stop Loss (pips) | 15, 20, 25, 30 |
| **tp\_pips** | 40 | Take Profit (pips) | 30, 40, 50, 60 |
| **martingale\_factor** | 1.3 | Hệ số nhân lot | 1.2, 1.3, 1.5 |
| **min\_adx** | 25 | ADX tối thiểu (filter sideway) | 20, 25, 30 |
| **min\_gap\_atr\_ratio** | 0.3 | FVG gap tối thiểu | 0.2, 0.3, 0.5 |
| **fvg\_lookback\_days** | 90 | FVG valid trong bao nhiêu ngày | 60, 90, 120 |

**6. ROADMAP PHÁT TRIỂN**

**PHASE 1: Python Backtest ⏳ [ĐANG THỰC HIỆN]**

**Milestone 1.1: FVG Module (Ưu tiên cao nhất)**

* ✅ fvg\_model.py - HOÀN THÀNH
* ✅ fvg\_detector.py - HOÀN THÀNH
* ⏳ fvg\_manager.py - ĐANG LÀM
* ⏳ fvg\_visualizer.py - TODO

**Timeline:** 1-2 ngày

**Milestone 1.2: Indicators Module**

* volume.py (OBV, CMF)
* momentum.py (MACD, RSI + divergence)
* volatility.py (ATR, Bollinger)
* trend.py (ADX)

**Timeline:** 2-3 ngày

**Milestone 1.3: Strategy Module**

* confluence\_scorer.py
* signal\_generator.py
* fvg\_confluence\_strategy.py (main)

**Timeline:** 1-2 ngày

**Milestone 1.4: Backtest Module**

* virtual\_engine.py
* real\_engine.py
* streak\_tracker.py
* backtester.py (orchestrator)

**Timeline:** 2-3 ngày

**Milestone 1.5: Data & Utils**

* data/download\_mt5\_data.py
* utils/data\_loader.py
* utils/metrics.py
* utils/logger.py
* config.py (hoàn thiện)

**Timeline:** 1 ngày

**Milestone 1.6: Testing & Validation**

* Unit tests (pytest)
* Chạy backtest trên 1 năm data
* Phân tích kết quả
* Optimize tham số

**Timeline:** 2-3 ngày

**Milestone 1.7: Documentation & Notebooks**

* README.md đầy đủ
* Jupyter notebook analysis
* Code comments
* API documentation

**Timeline:** 1 ngày

**🎯 TỔNG PHASE 1: 10-15 ngày**

**PHASE 2: MT5 Integration (Sau khi Phase 1 hoàn thành)**

**Milestone 2.1: Python → MQL5 Communication**

* Setup ZeroMQ hoặc Socket
* Python sender (tín hiệu)
* MQL5 receiver (EA)
* Test connection

**Timeline:** 2-3 ngày

**Milestone 2.2: MQL5 EA Development**

* Nhận tín hiệu từ Python
* Parse signal (BUY/SELL, SL, TP, lot)
* Đặt lệnh trên MT5
* Quản lý lệnh (modify, close)
* Send feedback về Python

**Timeline:** 3-4 ngày

**Milestone 2.3: Live Testing**

* Test trên demo account
* Monitor performance
* Fix bugs
* Paper trading 1-2 tuần

**Timeline:** 2-3 tuần

**🎯 TỔNG PHASE 2: 3-4 tuần**

**PHASE 3: AI Training (Sau khi có data từ Phase 1 & 2)**

**Milestone 3.1: Data Collection**

* Log đầy đủ:
  + Virtual & real trades
  + FVG history
  + Indicators values
  + Market conditions
* Chuẩn hóa data format
* Label data (setup good/bad)

**Timeline:** Tự động (trong Phase 1 & 2)

**Milestone 3.2: Feature Engineering**

* Chọn features quan trọng:
  + FVG strength, age
  + Confluence score breakdown
  + ADX, ATR values
  + Time of day, session
* Feature scaling/normalization
* Train/test split

**Timeline:** 3-5 ngày

**Milestone 3.3: Model Training**

* Thử các models:
  + Random Forest (baseline)
  + XGBoost
  + LSTM/GRU (time series)
  + Transformer (advanced)
* Hyperparameter tuning
* Cross-validation
* Evaluate metrics

**Timeline:** 1-2 tuần

**Milestone 3.4: Model Deployment**

* Export model (ONNX format)
* Integrate vào Python bot
* Backtest với AI predictions
* Compare: Rule-based vs AI

**Timeline:** 3-5 ngày

**🎯 TỔNG PHASE 3: 3-4 tuần**

**PHASE 4: Full MQL5 Migration (Optional)**

**Milestone 4.1: Rewrite Indicators in MQL5**

* FVG detection
* OBV, CMF, MACD, RSI, ATR, ADX
* Divergence detection

**Timeline:** 1-2 tuần

**Milestone 4.2: Embed AI Model in MQL5**

* ONNX runtime trong MQL5
* Load model
* Inference trong EA
* Optimize performance

**Timeline:** 1-2 tuần

**Milestone 4.3: Full EA**

* Standalone EA (không cần Python)
* All logic in MQL5
* Optimize & test
* Live deployment

**Timeline:** 1-2 tuần

**🎯 TỔNG PHASE 4: 3-6 tuần**

**7. SETUP & DEPLOYMENT**

**7.1. Cài đặt môi trường**

**Bước 1: Clone repository**

bash

git clone https://github.com/nguyenhongdat09/bot\_fvg\_ruled\_base.git

cd bot\_fvg\_ruled\_base/trading\_bot

**Bước 2: Tạo virtual environment**

bash

*# Windows*

python -m venv venv

venv\Scripts\activate

*# Linux/Mac*

python3 -m venv venv

source venv/bin/activate

**Bước 3: Install dependencies**

bash

pip install --upgrade pip

pip install -r requirements.txt

```

\*\*requirements.txt:\*\*

```

*# Data & Analysis*

pandas>=2.0.0

numpy>=1.24.0

MetaTrader5>=5.0.45

*# Technical Analysis*

pandas-ta>=0.3.14b

ta-lib>=0.4.28 *# Yêu cầu cài TA-Lib binary trước*

*# Visualization*

plotly>=5.14.0

matplotlib>=3.7.0

seaborn>=0.12.0

*# Backtesting*

backtrader>=1.9.78

*# Testing*

pytest>=7.3.0

pytest-cov>=4.0.0

*# Utilities*

tqdm>=4.65.0

loguru>=0.7.0

python-dotenv>=1.0.0

*# Future: ZeroMQ (Phase 2)*

*# pyzmq>=25.0.0*

*# Future: ML (Phase 3)*

*# scikit-learn>=1.2.2*

*# xgboost>=1.7.5*

*# tensorflow>=2.12.0 # hoặc pytorch*

*# onnx>=1.14.0*

*# onnxruntime>=1.15.0*

**Bước 4: Setup MT5 (nếu dùng MT5 data)**

1. Cài MetaTrader 5
2. Login demo/real account
3. Enable Algo Trading
4. Test connection:

python

import MetaTrader5 as mt5

if mt5.initialize():

print("✅ MT5 connected")

print(mt5.version())

mt5.shutdown()

else:

print("❌ MT5 connection failed")

**Bước 5: Download data**

bash

python data/download\_mt5\_data.py

**7.2. Chạy backtest**

bash

*# Single backtest*

python main.py --config config.py

*# Multiple tests (test matrix)*

python main.py --test-matrix

*# Specific test*

python main.py --test test\_1

*# With visualization*

python main.py --visualize

*# Debug mode*

python main.py --debug

**7.3. Cấu trúc main.py**

python

*# main.py*

"""

Main entry point for trading bot

"""

import argparse

from core.backtest.backtester import Backtester

from strategies.fvg\_confluence\_strategy import FVGConfluenceStrategy

from core.utils.data\_loader import load\_data

from core.utils.logger import setup\_logger

import config

def main():

*# Parse arguments*

parser = argparse.ArgumentParser(description='FVG Trading Bot')

parser.add\_argument('--test', type=str, help='Specific test name')

parser.add\_argument('--test-matrix', action='store\_true', help='Run all tests')

parser.add\_argument('--visualize', action='store\_true', help='Generate visualizations')

parser.add\_argument('--debug', action='store\_true', help='Debug mode')

args = parser.parse\_args()

*# Setup logger*

logger = setup\_logger(debug=args.debug)

logger.info("🚀 Starting FVG Trading Bot")

*# Load data*

logger.info(f"📊 Loading data: {config.DATA\_CONFIG['symbol']} {config.DATA\_CONFIG['timeframe']}")

data = load\_data(config.DATA\_CONFIG)

logger.info(f"✅ Data loaded: {len(data)} candles")

*# Initialize strategy*

strategy = FVGConfluenceStrategy(config.STRATEGY\_CONFIG)

if args.test\_matrix:

*# Run multiple tests*

logger.info("🧪 Running test matrix...")

results = {}

for test\_name, test\_config in config.TEST\_MATRIX.items():

logger.info(f"\n{'='\*60}")

logger.info(f"Test: {test\_name}")

logger.info(f"Config: {test\_config['description']}")

logger.info(f"{'='\*60}\n")

backtester = Backtester(strategy, test\_config)

result = backtester.run(data)

results[test\_name] = result

*# Compare results*

compare\_results(results)

else:

*# Single test*

test\_config = config.TEST\_MATRIX.get(args.test, config.TEST\_MATRIX['test\_1'])

logger.info(f"🧪 Running test: {args.test or 'default'}")

logger.info(f"Config: {test\_config['description']}")

backtester = Backtester(strategy, test\_config)

result = backtester.run(data)

*# Print results*

print\_results(result)

*# Visualize*

if args.visualize:

logger.info("📈 Generating visualizations...")

generate\_visualizations(result, data)

logger.info("✅ Done!")

def compare\_results(results):

"""So sánh kết quả các tests"""

*# Implementation*

pass

def print\_results(result):

"""In kết quả backtest"""

*# Implementation*

pass

def generate\_visualizations(result, data):

"""Tạo charts"""

*# Implementation*

pass

if \_\_name\_\_ == "\_\_main\_\_":

main()

**8. TESTING & VALIDATION**

**8.1. Unit Tests**

bash

*# Run all tests*

pytest tests/

*# Specific module*

pytest tests/test\_fvg\_detector.py

*# With coverage*

pytest --cov=core tests/

**test\_fvg\_detector.py example:**

python

*# tests/test\_fvg\_detector.py*

import pytest

import pandas as pd

from core.fvg.fvg\_detector import FVGDetector

def test\_detect\_bullish\_fvg():

"""Test phát hiện Bullish FVG"""

*# Mock data*

data = pd.DataFrame({

'high': [1.1000, 1.1020, 1.1100],

'low': [1.0980, 1.1000, 1.1080]

})

detector = FVGDetector(min\_gap\_atr\_ratio=0.3)

atr = 0.0050

fvg = detector.detect\_fvg\_at\_index(data, index=2, atr=atr)

assert fvg is not None

assert fvg.fvg\_type == 'BULLISH'

assert fvg.bottom == 1.1020

assert fvg.top == 1.1080

def test\_detect\_bearish\_fvg():

"""Test phát hiện Bearish FVG"""

*# Similar implementation*

pass

def test\_gap\_too\_small():

"""Test gap quá nhỏ bị reject"""

*# Implementation*

pass

**8.2. Integration Tests**

python

*# tests/test\_integration.py*

def test\_full\_backtest\_flow():

"""Test toàn bộ flow backtest"""

*# Load data*

*# Initialize components*

*# Run backtest*

*# Verify results*

assert result['virtual\_trades\_count'] > 0

assert result['real\_trades\_count'] > 0

assert result['metrics']['sharpe\_ratio'] > 0

**8.3. Validation Checklist**

**FVG Module:**

* FVG được phát hiện chính xác (không lookahead)
* FVG tracking real-time đúng
* FVG bị chạm → mất hiệu lực ngay
* FVG > 90 ngày → bị remove
* Market structure logic đúng

**Indicators:**

* ATR, ADX tính toán đúng
* Divergence detection chính xác
* OBV, CMF values hợp lý
* Bollinger Bands không lag

**Strategy:**

* Confluence score tính đúng
* Signal generation logic chính xác
* FVG validation đúng QUY TẮC VÀNG

**Backtest:**

* Virtual trades log đầy đủ
* Real trades với martingale đúng
* Mode switching đúng logic
* Metrics tính toán chính xác

**9. TROUBLESHOOTING**

**9.1. Common Issues**

**Issue 1: MT5 không kết nối được**

python

*# Fix:*

*# 1. Check MT5 đã mở chưa*

*# 2. Check Algo Trading enabled*

*# 3. Restart MT5*

*# 4. Reinstall MetaTrader5 package*

pip uninstall MetaTrader5

pip install MetaTrader5

**Issue 2: FVG không được phát hiện**

python

*# Check:*

*# 1. min\_gap\_atr\_ratio có quá cao?*

*# 2. ATR có = 0 không?*

*# 3. Data có đúng format không? (OHLC)*

*# Debug:*

detector = FVGDetector(min\_gap\_atr\_ratio=0.1) *# Giảm threshold*

**Issue 3: Backtest chạy chậm**

python

*# Optimize:*

*# 1. Giảm lookback\_candles*

*# 2. Vectorize calculations*

*# 3. Dùng numba @jit*

*# 4. Reduce logging level*

import logging

logging.basicConfig(level=logging.WARNING) *# Thay vì INFO*

**Issue 4: Memory error với data lớn**

python

*# Fix:*

*# 1. Chunk data thành nhiều đoạn*

*# 2. Dùng generator thay vì load all*

*# 3. Giảm precision (float32 thay float64)*

data = data.astype('float32')

**9.2. Debug Tips**

**Enable verbose logging:**

python

*# core/utils/logger.py*

import logging

logging.basicConfig(

level=logging.DEBUG,

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s'

)

**Print intermediate values:**

python

*# Trong backtest loop*

if i % 100 == 0:

print(f"Candle {i}/{len(data)}")

print(f"FVG count: {len(fvg\_manager.active\_bullish\_fvgs) + len(fvg\_manager.active\_bearish\_fvgs)}")

print(f"Mode: {mode}")

print(f"Streak: {virtual\_engine.loss\_streak}")

**Visualize problematic cases:**

python

*# Khi signal bị reject, vẽ chart*

if signal is None and confluence\_score >= 7:

visualizer.plot\_debug\_chart(data, fvg\_manager, indicators)

**10. APPENDIX**

**10.1. Glossary**

| **Term** | **Định nghĩa** |
| --- | --- |
| **FVG** | Fair Value Gap - Gap giữa 3 nến liên tiếp |
| **Confluence** | Sự trùng khớp của nhiều chỉ báo |
| **Divergence** | Phân kỳ giữa price và indicator |
| **Lookahead Bias** | Lỗi dùng data tương lai trong backtest |
| **Martingale** | Tăng lot sau khi thua để recover |
| **Drawdown** | Tỷ lệ % giảm từ đỉnh equity |
| **Sharpe Ratio** | Tỷ lệ return/risk |
| **Profit Factor** | Tổng profit / Tổng loss |

**10.2. References**

1. **FVG Concept:**
   * ICT (Inner Circle Trader) methodology
   * Smart Money Concepts
2. **Technical Indicators:**
   * Investopedia
   * TradingView documentation
3. **Python Libraries:**
   * pandas-ta: <https://github.com/twopirllc/pandas-ta>
   * TA-Lib: <https://mrjbq7.github.io/ta-lib/>
4. **MT5 API:**
   * <https://www.mql5.com/en/docs/integration/python_metatrader5>

**10.3. Contact & Support**

* **GitHub:** <https://github.com/nguyenhongdat09/bot_fvg_ruled_base>
* **Issues:** GitHub Issues
* **Documentation:** README.md + code comments

**📝 SUMMARY - ĐIỂM QUAN TRỌNG CẦN NHỚ**

**🎯 Core Logic:**

1. **FVG chạm = mất hiệu lực** (không có lấp 50%)
2. **FVG lookback 90 ngày** (calendar days)
3. **QUY TẮC VÀNG:**
   * BUY chỉ khi có FVG dưới (hoặc có cả 2)
   * SELL chỉ khi có FVG trên (hoặc có cả 2)
   * Không có FVG = NO TRADE

**🔢 Tham số mặc định:**

* Loss streak trigger: **3**
* Min confluence: **7/14 điểm**
* SL: **20 pips**, TP: **40 pips** (RR 1:2)
* Martingale: **1.3x**
* ADX min: **25**

**📊 Modules ưu tiên:**

1. ✅ FVG (model + detector) - DONE
2. ⏳ FVG (manager + visualizer) - IN PROGRESS
3. ⏳ Indicators - TODO
4. ⏳ Strategy - TODO
5. ⏳ Backtest - TODO

**🚀 Next Steps:**

1. Hoàn thiện **fvg\_manager.py**
2. Code **fvg\_visualizer.py**
3. Implement **indicators module**
4. Build **strategy module**
5. Complete **backtest engine**