🛡️ Dynamic Shield v3.0 Implementation Master Plan

목표: K-ICS 연계형 AI 환헤지 솔루션 프로토타입 개발 및 검증

환경: Local macOS (VS Code + Jupyter), Python 3.10+

핵심 전략: 1. Architecture Emulation: Kafka/Redis 대신 Queue/Threading으로 비동기 논리 구현

2. Data Augmentation: 꼬리 위험(Tail Risk) 학습을 위한 가상 폭락장 생성

3. Solvency Focus: 단순 수익률이 아닌 'K-ICS 비율 방어' 입증

📅 Phase 1: The Engine (기반 구축)

핵심 논리**:** 실제 K-ICS 엔진이 없으므로, 제안서의 수식을 완벽히 모사한 **'Ground Truth(정답지)'**를 먼저 만들어야 함 [1].

1.1. High-Fidelity Data Acquisition (데이터 수집)

- **Action:** 터미널에서 엑셀(Excel)로 데이터를 내려받은 후 CSV로 저장하여 파이썬으로 로드. (API 연동보다 CSV Export가 7일 컷에는 훨씬 빠르고 안전함)
- **Data Points (Tickers Example):**
    - **Global (Bloomberg):**
        - US Treasury 10Y Yield: USGG10YR Index (미국채 10년물 금리 - 필수)
        - S&P 500: SPX Index
        - FX Spot: KWN Curncy (원/달러 환율)
        - **[중요] Swap Points:** KWN1M Curncy (1개월물 스왑포인트) 또는 1년물. 헤지 비용 계산의 핵심.
        - Crisis Indicators: VIX Index (공포지수), KORCDS Index (한국 CDS 프리미엄 - 평가서에서 강조한 선행지표 [2])
    - **Domestic (FnGuide):**
        - Equity: KOSPI 200 지수 (구성종목 말고 지수 자체 사용)
        - Rates: 국고채 10년 금리 (K-ICS 금리 리스크 산출용)

1.2. Data Pre-processing (전처리)

- **Merge:** 서로 다른 소스(Bloomberg vs FnGuide)의 날짜(Date)를 기준으로 Inner Join.
    - 주의: 한국 휴장일과 미국 휴장일이 다르므로, 데이터가 비어 있는 날은 Forward Fill(ffill)로 전일 종가를 채워 넣어야 데이터가 안 끊김.
- **Derive Variables:**
    - **Hedge Cost (헤지 비용):** Swap Point / FX Spot (이 값이 마이너스면 헤지할수록 손해라는 논리 증명 가능)
    - **Risk-Free Rate:** 미국채 10년물 금리를 무위험 이자율로 사용.

1.3. Rule-based K-ICS Calculator (kics_real.py)

- **Action:** 제안서의 제곱근 합산 수식 구현 (이전과 동일).
- **Upgrade:** 이제 정확한 **'국고채 금리'**와 **'미국채 금리'**가 있으므로, **금리 리스크(Interest Rate Risk)**까지 수식에 포함할 수 있음.

$$TotalRisk = \sqrt{Risk_{Equity}^2 + Risk_{FX}^2 + Risk_{Rate}^2 + (Correlations...)}$$

    - 심사위원에게 **"실제 국고채/미국채 금리 스프레드까지 반영하여 K-ICS 금리 리스크를 정교하게 산출했다"**고 어필 가능.
-

📅 Phase 2: The Brain (모델링)

핵심 논리: 느린 연산 속도를 극복하기 위한 **Surrogate Model**과 시장 비정상성을 다루는 **Regime Detection** 구현 [5].

2.1. Tail Risk Data Augmentation (데이터 증강)

- **Why:** 과거 데이터만으로는 위기 상황 학습 부족 [6].
- **Action:** kics_real.py를 활용해 몬테카를로 시뮬레이션 수행.
    - 가상 시나리오: 주가 -30% & 환율 +20% 동시 발생 데이터 1만 개 생성.
    - 이 데이터를 학습셋에 포함시켜야 AI가 위기 때 멍청해지지 않음.

2.2. Deep Neural Surrogate (kics_surrogate.py)

- **Structure:** Input(시장변수) -> 3-Layer DNN (ReLU/ELU) -> Output(K-ICS Ratio) [7].
- **Validation:** K-ICS 비율 100%~150% 구간(위험 구간)에서의 오차율 1% 미만 달성 필수 [8].

2.3. Regime Detection (regime.py)

- **Logic:** HMM (Hidden Markov Model) 활용 3단계 분류 (Normal, Panic, Transition) [9].
- **Feature:** VIX, 이동평균 괴리율, **CDS** 프리미엄(또는 대체 지표) 포함하여 선행성 확보 [10].

📅 Phase 3: The System (아키텍처)

핵심 논리: **Event-Driven MSA**와 **Fast/Slow Track** 분리 구조를 로컬 파이썬으로 모사 [11].

3.1. Pseudo-Async Architecture

- **Tool:** Python threading, queue.
- **Structure:**
    1. **Main Thread (Market Data Stream):** 데이터를 큐에 계속 put().
    2. **Fast Track Thread (Inference):** 큐에서 get() -> Surrogate 모델로 즉시 헤지 비율 산출 -> 로그 출력.
    3. **Slow Track Thread (Learning):** (백그라운드) 100틱마다 실제 엔진(kics_real) 호출하여 Surrogate 오차 보정 학습.
- **Defense Logic:** "운영 복잡도를 줄이기 위해 인메모리 큐 방식으로 아키텍처의 논리적 구조만 모사했습니다."

📅 Phase 4: The Agent (에이전트)

핵심 논리: **자본 최적화(Solvency Optimization)**를 위한 강화학습 및 안전장치**(Safety Layer)** [12].

4.1. RL Environment Setup

- Objective: Reward Function R_t 구현이 핵심.

$$R_t = (r_{port} - r_{bm}) - \lambda_1 |Transaction| - \lambda_2 \cdot Penalty(KICS < 100\%)$$

- **Constraint:** K-ICS 비율이 100% 미만으로 떨어지면 λ_2 페널티를 아주 강력하게 부여 (-1000점) [13].

4.2. Safety Layer (Gradual De-risking)

- **Logic:** AI가 패닉에 빠지거나 급발진하는 것을 막는 하드코딩 룰 [14].

Implementation: agent.py의 get_action() 메서드 내부에 구현.
Python

```
if regime == 'PANIC' and VIX > 30:
    target_hedge = calculate_min_hedge_for_safety()
    current_hedge = move_slowly(current_hedge, target_hedge, step=0.05)
    return current_hedge
```

📅 Phase 5: Verification (검증 및 시각화)

핵심 목표: 단순 수익률**(CAGR)** 경쟁이 아닌, **'위기 시 자본 방어 능력**(Solvency Defense)'**과 **'리스크의 역설**(Risk Paradox)'**을 수치와 그래프로 입증하여 심사위원을 설득함.

5.1. Risk Paradox Proof (이론적 타당성 검증)

- **Target: "**헤지를 덜 했는데**(80~90%)** 왜 총 위험액**(Total Risk)**은 줄어드는가**?"**에 대한 수학적 증명 .
- **Action:** `src/validation/proof_risk_paradox.py` 실행.
- **Check Point:**
  - **[ ]** 헤지 비율 **100%**일 때보다 **80%**일 때의 **K-ICS** 총 위험액이 더 작게 산출되는지 확인.
  - **[ ]** 콘솔에 `[SUCCESS] Risk Paradox Proven!` 메시지가 출력되는지 확인.
- **Significance:** 주식**-**환율 음의 상관관계**(Natural Hedge)**로 인한 분산 효과를 수치로 입증하여 규제적 타당성 확보 .

5.2. Solvency Analysis (자본 적정성 시각화**)**

- **Target: 2020**년 **3**월 코로나 팬데믹**(**주가 폭락 **&** 환율 급등**)** 시나리오 시뮬레이션 .
- **Action:** `src/validation/solvency_visualizer.py` 실행.
- **Plot: X**축**(**시간**), Y**축**(K-ICS** 비율**)** 그래프 생성 **(**`kics_defense_result.png`**).**
  - **Line A (Benchmark 100% Hedge):** 스왑 비용 누적과 마진콜 발생으로 **K-ICS** 비율 급락 .
  - **Line B (Dynamic Shield):** 환차익을 활용한 자연 헤지 효과로 **K-ICS** 비율이 안정권**(150%** 이상**)** 유지 .
- **Significance:** 이 그래프가 ******제안서의 가장 강력한 무기**(The Money Shot)****임 .

5.3. Safety Layer Stress Test (시스템 안전성 검증**)**

- **Target: AI** 오작동이나 극단적 공포 국면**(VIX > 40)**에서의 강제 제어 시스템 작동 확인 .
- **Action:** `src/validation/stress_safety.py` 실행.
- **Check Point:**
  - **[ ] VIX 40** 이상 데이터를 강제 주입했을 때**,** 시스템 로그에 ****"Emergency: Gradual De-risking Triggered"****가 출력되는지 확인.
  - **[ ]** 헤지 비율이 급격하게 변하지 않고**,** 설정된 **Step(**예**: 0.05)** 단위로 천천히 상승하는지 확인 .

5.4. Walk-Forward Backtesting (성과 분석**)**

- **Target:** 미래 데이터를 미리 보지 않도록**(Look-ahead Bias** 제거**) Rolling Window** 방식으로 검증 .
- **Comparisons:** 다음 **3**가지 전략의 **CAGR(**수익률**)** 및 **MDD(**최대 낙폭**)** 비교 .
  1. **Benchmark (100% Hedge)**
  2. **Static Partial (80% Fixed Hedge)**
  3. **Dynamic Shield v3.0 (AI)**

5.5. Advanced Visualization (수상을 위한 심화 시각화**: XAI & Efficient Frontier) [Updated]**

- **Target:** 단순 성과**(CAGR)**를 넘어**, **'AI** 판단의 투명성**(Why Not)'****과 ******'비용 대비 효율성**(Cost-Efficiency)'****을 시각적으로 증명하여 심사위원의 신뢰 획득.
- **Action:** `src/validation/advanced_viz.py` 실행 **(**로직 고도화**).**

- **Plot 1 (Counterfactual Dashboard - Why Not 분석):**
  - **X축(VIX 또는 환율 변동성), Y축(헤지 비율)**으로 구성된 **'의사결정 경계선(Decision Boundary)'** 그래프 생성.
  - **Check Point: "현재는 헤지 비율 80%**를 유지 중이나**,** 만약 **VIX**가 **10%** 추가 상승했다면 **AI**는 즉시 **95%**로 비율을 상향했을 것**"**임을 보여주는 시뮬레이션 궤적**(Trajectory)**이 시각화되는지 확인**.**
- **Plot 2 (Efficient Frontier - 효율적 투자선):**
  - **X축(Total Risk), Y축(Total Hedge Cost)** 산포도 작성**.**
  - **Check Point: Dynamic Shield**의 좌표가 **Benchmark(100%** 헤지**)** 대비 ****'**좌측 하단**(Low Risk, Low Cost)'****의 **Sweet Spot**에 위치하여**,** 리스크와 비용을 동시에 낮췄음을 한눈에 입증하는지 확인**.**

📅 Phase 6: Final Review & Packaging (최종 점검)

6.1. Logic Consistency Check (기존 논리 점검)

- **Risk Paradox** 증명**:** 백테스팅 결과, 헤지 비율 80~90% 구간에서 주식-환율 음의 상관관계로 인해 총 위험액**(Total Risk)**이 오히려 감소함을 확인했는가?
- **Safety Layer** 작동**:** VIX 40 이상 데이터 주입 시, 로그에 **"Emergency: Gradual De-risking Triggered"**가 출력되며 헤지 비율이 천천히 상승하는가?
- **Surrogate** 오차**:** K-ICS 비율 100% 근처(위험 구간)에서 Surrogate 모델과 Real 모델 간의 오차율이 허용 범위 내인가?

6.2. Award-Winning Action Items (수상을 위한 필살기) [NEW]

- **'Why Not'** 분석 시각화**:** SHAP 리포트를 통해 "AI가 왜 100% 헤지를 제안하지 않았는가?"에 대한 답변(예: 금리차 확대, 스왑 포인트 마이너스 심화 등)을 데이터로 증명했는가?
- 효율적 투자선**(Efficient Frontier)** 도출**:** 산포도를 통해 본 솔루션이 리스크와 비용을 동시에 낮추는 **'최적점(Sweet Spot)'**을 찾아냈음을 한눈에 보여주는가?

- 비용 효율성 지표**(RCR)** 강조**:** 단순 수익률 외에 **RCR(Risk-Cost Ratio)** 지표를 제시하여, 1원의 비용으로 얼마의 자본을 절감했는지 수치로 보여주는가?
- 코드 주석의 철학적 통일**:** 제출하는 모든 코드의 Docstring과 주석에 **"Capital Optimization, not Prediction (환율 예측이 아닌 자본 최적화)"**라는 개발 철학을 명시하여, 심사위원이 단순 봇으로 오해할 여지를 원천 차단했는가?