



# AI Strategy Engine – Gold & Silver





## Extension for Bot Builder ProRealTime

Domain-specific AI for probabilistic trading strategies on **Silver (XAG)** and **Gold (XAU)**

---

## Overview






The **AI Strategy Engine** is an advanced extension to the **Bot Builder ProRealTime** project. It introduces a domain-locked AI layer specialised exclusively in **gold and silver markets**, producing:

-  Multi-timeframe directional bias
-  **Price targets with probability (%)**
-  **Probability curves** (distribution-based, not point forecasts)
-  **Guaranteed-valid ProRealTime (PRT) trading code**

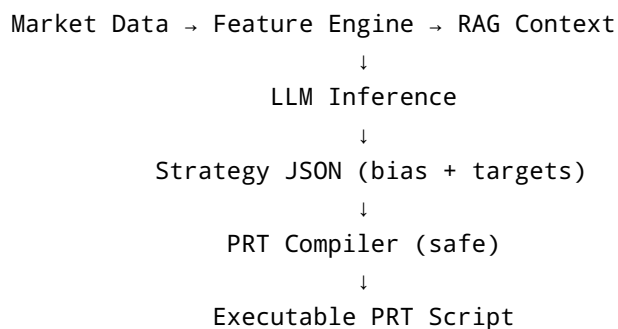
This is **not** a black-box price predictor. It is a **market-state interpreter** → **strategy generator** → **PRT compiler**.

---

## Key Principles

-  No raw price prediction by LLM alone
  -  Regime, structure, and session awareness
  -  Probabilistic outputs instead of certainties
  -  Deterministic, execution-safe PRT generation
  -  Explainable, auditable logic
- 

## Architecture



# Technology Stack

## Frontend

- **Vite + React**
- Integrated as a **new tab** in the existing Bot Builder UI

## Backend

- **Node.js (Express)** – API orchestration
- **Java service** – feature engineering & regime logic

## AI Layer

- **Mistral 7B Instruct** (recommended)
- Fine-tuning via **LoRA / QLoRA**
- **RAG (Retrieval-Augmented Generation)** for historical grounding

## Storage

- Market data cache (OHLCV)
  - Vector database (FAISS / Chroma)
- 

## UI – AI Strategy Engine Tab

### Controls

- Symbol:  /
- Timeframes:  ·  ·  ·  ·  ·
- Session filter:  /  /
- Mode:  ·  ·

### Output Panels

1. **Market State Summary**
  2. **Directional Bias (per timeframe)**
  3. **Price Target Probability Curves**
  4. **Trade Logic Explanation**
  5. **Generated ProRealTime Code**
- 

## Market Feature Engine (Java)

### Responsibilities

- Session classification

- Volatility regime detection
- Liquidity estimation
- Multi-timeframe structure alignment
- Feature normalization

## Example Output

```
{
  "session": "London",
  "regime": "LowLiquidityFakeout",
  "volatility": "Elevated",
  "structure": {
    "M5": "Reversal",
    "H1": "Range"
  }
}
```

## AI Output Contract (Strict JSON)

### Directional Bias

```
{
  "bias": {
    "M5": "long",
    "M15": "long",
    "H1": "neutral"
  }
}
```

## Price Targets & Probabilities

Targets are **zones**, not exact prices. Each target includes:

- Relative move ( % )
- Probability ( 0-1 )
- Timeframe context

### Example

```
{
  "targets": {
```

```
"M5": [
  { "move": "+0.15%", "probability": 0.62 },
  { "move": "+0.30%", "probability": 0.41 },
  { "move": "-0.10%", "probability": 0.22 }
],
"H1": [
  { "move": "+0.45%", "probability": 0.38 },
  { "move": "-0.35%", "probability": 0.29 }
]
}
}
```

---

## Probability Curves

Probability curves are derived from:

- Historical volatility
- Session-specific behaviour
- Similar historical regimes (via RAG)

### Internal Representation

```
{
  "distribution": "skewed-normal",
  "skew": "upside",
  "confidence": 0.71
}
```

Displayed in the UI as bell / skew curves and cumulative probability bands.

---

## RAG (Retrieval-Augmented Generation)

### Vector Store Contains

- Historical gold & silver events
- Flash crash cases
- Session behaviour notes
- Past strategies & outcomes
- Existing PRT scripts

## Flow

Market Snapshot → Embedding → Similar Cases → Prompt Enrichment → Inference

This anchors the AI in **real market behaviour** and prevents hallucinations.

---

## ProRealTime Compiler (Node.js)

### Input Schema

```
{
  "entry": "London fake breakdown reversal",
  "direction": "long",
  "stop": { "type": "ATR", "value": 1.2 },
  "target": "+0.30%",
  "timeframe": "M5"
}
```

### Output (Guaranteed Valid PRT)

```
DEFPARAM CumulateOrders = False

IF NOT OnMarket THEN
  IF Close CROSSES OVER VWAP THEN
    BUY 1 CONTRACT AT MARKET
  ENDIF
ENDIF

SET STOP pLOSS ATR(14) * 1.2
```

Compiler rules enforce: - Valid PRT syntax - Supported indicators only - Deterministic behaviour

---

## Repository Extension Layout

```
Bot_builder_ProRealTime/
├─ tabs/
│   ├─ manual-builder
│   ├─ indicators
│   └─ ai-strategy-engine
│       └─ StrategyUI.tsx
```

```
|      |─ ProbabilityCurves.tsx
|      └─ TargetTable.tsx
|─ server/
|   └─ market-engine
|   └─ ai-gateway
|   └─ prt-compiler
|   └─ rag-service
```

---

## Training Data Format (LoRA)

```
{
  "context": "Silver London open low-liquidity breakdown",
  "features": {
    "session": "London",
    "volatility": "High"
  },
  "decision": {
    "bias": "Long",
    "target": "+0.25%"
  },
  "outcome": "Success"
}
```

Focus on **quality over quantity**.

---

## Safety & Constraints

- Trade frequency caps
  - Session-based limits
  - Hard risk constraints
  - Timeframe agreement checks
- 

## Recommended Development Order

1. Define JSON contracts
2. Implement PRT compiler
3. Build market feature engine
4. Add RAG layer
5. Integrate LLM inference
6. Fine-tune last

---

## Result

You get a **professional-grade AI strategy engine** that is:

- Silver & gold only
- Session-aware
- Probability-driven
- Explainable
- ProRealTime-safe

This is designed for **real trading systems**, not demos.