Intraday Predictive Modeling & Execution Framework — Research Specification

1. Research Goal

You are provided with intraday feature sets (one file per trading day).
Your objective is to design a **causal, iteration-safe prediction engine** and a **fully executable trading strategy** that operates only on **P3**, with realistic costs.

The core requirement:

→ Predict the future behavior of P3 at a minimum lookahead of 30 bars (30-row horizon) .

This ensures the model is truly forward-looking and avoids trivial near-term noise-chasing.

The output of your research is:

1. A clearly explained **modeling rationale**
2. A reproducible, well-documented **predictive framework**
3. A fully iterative **execution engine** that generates positions & PnL
4. Daily-level trade logs with proper cost accounting (0.01% TC)
5. You can choose any horizon greater than 30 bars

---

2. Dataset Structure

Each file (e.g., `1.csv`, `2.csv`, …) corresponds to a **single trading day** of intraday observations.

Key Columns

- `ts` — Timestamp (use for strict ordering).
- `P3` — Tradeable mid/price series used for execution.

- Optional price proxies: `P1`, `P2`, `P4`.
- A large set of engineered features.

Feature Naming Convention

Feature names contain hierarchical information via `_`.

Example:
`F_H_B` → tokens: **F**, **H**, **B**

Researchers may exploit this structure for:

- Group-level statistics
- Cross-group interactions
- Horizon-wise aggregates
- Regime features
- Family-wise normalization

You are free to design additional derived features as long as all processing is causal.

---

3. Modeling Requirements

Your modeling pipeline should include:

1. **Feature sanitation**
2. **Feature engineering**
3. **Model selection**
    - Can be tree-based (GBDT/XGB/LGBM), linear, logistic, or hybrid.
    - Must be trained **offline** or using a safe per-day expanding window .

4. **Causality**
    - No target leakage
    - No forward-looking windows
    - No using future rows for normalization inside the day

You are allowed to pretrain on historical days, but **within-day execution must be iterative**.

---

4. Execution Layer (Trading Strategy)

The execution engine acts as a **minimal-position strategy** operating on `P3`.

Signals

- `+1` = long
- `-1` = short
- `0` = flat

Signal logic must be derived directly from model predictions.

Causal Iteration

For each timestamp `t`:

1. Read features up to `t`
2. Produce model output for horizon t → t+30 or more (predict for next 30 seconds or more)
3. Convert output into trading signal
4. Execute entry/exit logic on P3

5. Update PnL with **0.01% transaction cost**

Transaction Cost Model (0.01%)

- Deduct costs on:
  - Long ↔ Short flips
  - Entries
  - Exits
- Must be applied based on executed price at time `t`.

PnL Accounting

Track:

- Position
- Entry price
- Realized PnL
- MTM PnL
- Transaction costs
- Cumulative PnL

Final output: a per-day trade log.

---

5. Deliverables

A. Modeling Write-up (Quant Theory Doc)

Short internal-style note covering:

- Motivation for forward horizon

- Feature families based on _ structure
- Target design
- Model choice + reasoning
- Cross-validation / walk-forward logic
- Conversion of predictions → trade signals
- Risk considerations (overfitting, stationarity, normalization)

B. Runnable Code (Research → Production Bridge)

A script:

```
python strategy.py --input day.csv --output trades_day.csv
```

C. Documentation

- Docstrings explaining each module
- Clear comments on:
  - Causality enforcement
  - Rolling window logic
  - Signal formation
  - Cost deduction
- Short README with usage instructions

---

6. Style Expectations

Your solution should reflect:

- Clean, readable architecture
- Causality-first thinking

- Reproducibility
- Execution realism
- Proper separation between modeling and trading layers
- Solid understanding of microstructural noise vs actionable signal