ORB Intraday Meta-Model System – Technical Specification

> **Purpose** — Automate data ingest, feature engineering, model tuning/selection, walk-forward evaluation, and daily inference for an Opening-Range-Breakout (ORB) trading strategy. Execution logic will be added later; this phase stops at a validated blotter CSV.

---

1 · High-Level Workflow

1. **Nightly ingest** 1-min bars (Polygon) → Parquet.
2. **Morning (≈ 09 : 05 ET)** pull Barchart screener list (optional).
3. **Feature builder** creates opening-range + context features and forward-return label.
4. **Meta-model training** on rolling 250-session window, re-trained every 40 sessions.
5. **Hyper-tuning & CV** via Optuna; experiments logged to MLflow.
6. **Walk-forward evaluator** simulates 10 : 00 entries and 15 : 55 exits; metrics stored.
7. **Report generator** produces Markdown/PDF with SHAP plots & stats.
8. **Live inference** reads `/models/latest` and outputs a blotter CSV.

---

2 · Repository Layout

```
orb_system/
├─ pyproject.toml      # poetry; pins pandas, polars, lightgbm, torch, hydra-
core, mlflow, optuna
├─ config/
│   ├─ core.yaml       # API keys & data paths
│   ├─ train.yaml      # model + CV grids
│   └─ assets.yaml     # core basket & screener filters
├─ orb/                # importable package
│   ├─ data/           # polygon_loader.py, feature_builder.py,
barchart_scanner.py
│   ├─ models/         # base_model.py, lgbm_model.py, tcn_model.py, ...
│   ├─ tuning/         # tuner.py (Optuna / RayTune)
│   ├─ evaluation/     # walkforward.py, backtester.py
│   ├─ reporting/      # shap_utils.py, report_maker.py
│   ├─ cli/            # __init__.py (Typer commands)
│   └─ utils/          # calendars.py, logging.py
├─ data/               # auto-created; raw/, minute/, feat/
├─ models/             # timestamped snapshots + latest symlink
├─ mlruns/             # MLflow experiments
├─ scripts/            # cron/Airflow shims (retrain_if_due.sh, daily_scan.sh)
└─ notebooks/          # ad-hoc EDA / SHAP deep dives
```

---

3 · Data Sources & Formats

Source	Endpoint	Saved As
Polygon	`/v2/aggs/ticker/{sym}/range/1/minute/{start}/{end}`	`data/raw/{sym}/{yyyymm}.json.gz`
Polygon	**Minute Parquet**	`data/minute/{sym}.parquet`
Barchart	`getStocks.json` filter URL	Screener JSON cached per-day

All timestamps **UTC → America/New_York** before writing Parquet.

---

4 · Feature Engineering (feature_builder.py)

Name	Formula
`or_high / low / range / vol`	max/min/sum 09∶30–10∶00
`atr14pct`	14-day ATR ÷ last close
`ema20_slope`	pct change of 20-EMA over last bar
`vwap_dev`	last close − session VWAP
label `y`	3-hour forward return > 0 ⇒ 1 else 0

Forward-fill higher-TF columns to 1-min index so no look-ahead.

---

5 · Model Block

- `BaseModel` ABC → `.fit()  .predict_proba()  .save()  .load()`
- Implementations: `LGBMModel`, optional `TCNModel` (PyTorch).
- `MetaEnsemble` stacks any N base models + meta learner (log-reg or LightGBM).

---

6 · Training, CV & Walk-Forward

- **Optuna** search space defined in `config/train.yaml`.
- **TimeSeriesSplit** inside Optuna trial.

- **walkforward.py**
- rolling 250-session train / 25-session val
- re-fit every 40 sessions
- logs precision, recall, Sharpe, max DD to MLflow.

---

7 · Retrain Cadence (scripts/retrain_if_due.sh)

```
python -m orb.cli.retrain_if_due --train-window 250 --retrain-gap 40 \
       --precision-floor 0.55 --sharpe-floor 0.7
```

- Success → snapshot `models/meta_orb_YYYYMMDD/` + update `models/latest`.

---

8 · Daily Scan & Score

```
python -m orb.cli.scan_and_score --date 2025-07-15 \
       --scanner-config config/assets.yaml \
       --model-path models/latest \
       --out blotters/20250715.csv
```

Outputs: `symbol, side, prob, or_range, tp_raw, sl_raw` (no execution yet).

---

9 · Reporting

- `orb.reporting.report_maker.generate(exp_id, out_path)` → Markdown / PDF
- Includes equity curve, confusion matrix, SHAP bar, top interactions.

---

10 · Cursor Prompt (paste into Cursor AI)

```
You are acting inside the *orb_system* repository.  Please create the following:
1. Python package `orb` with sub-packages: `data`, `models`, `tuning`,
`evaluation`, `reporting`, `cli`, `utils`.
2. Implement `orb/utils/calendars.py` with NYSE session helpers (`trading_days`,
`nth_prev_session`, `is_retrain_day`).
3. Implement `orb/data/polygon_loader.py` with a `download_month(sym, yyyy, mm)`
function that saves raw JSON and a `build_parquet(sym)` that writes consolidated
Parquet.
4. Implement `orb/data/feature_builder.py` containing `build_features(sym, date,
minute_df)` returning a pandas.Series with the columns listed in section 4.
```

```
5. Implement `orb/models/base_model.py` (ABC) and `orb/models/lgbm_model.py`
extending it.
6. Implement `orb/cli/train.py` (Typer command) that:
   • loads 250-session window,
   • runs Optuna with search space from `config/train.yaml`,
   • logs to MLflow,
   • saves best model under `/models/meta_orb_<date>/`.
7. Add Hydra config files described in `config/`.
Follow PEP-8, add type hints, and write clear docstrings.
```

How to Use This Doc

- Keep it in the repo root (or `/docs/`). Cursor AI and teammates will reference it.
- Edit `config/*.yaml` rather than hard-coding paths in code.
- Each future module request in Cursor can say: “see ORB_System_Spec §X for details.”