Slice: A High-Level Architectural Overview

---

1. What Slice (or Meridian) Actually Is

Slice is a **personal macro-investment system** designed to enhance Liam's thinking, structure his investment theses, and automate parts of the portfolio that should *not* require daily attention.

It is **not**:

- a trading bot
- a forecasting model
- an AI trader
- an LLM that hallucinates narratives

Instead, Slice is:

A disciplined macro process made executable.

It combines:

1. **Structured thesis objects** (ThesisJSON)
2. **Quant checks** (backtests and data conditioning)
3. **LLM support** (parsing, critique, summarization — *never execution*)
4. **Portfolio autonomy** (bands, stops, rebalancing, timed actions)
5. **Daily monitoring** (Morning Briefing + drift alerts)
6. **A database backbone** (Postgres + pgvector)
7. **A CLI and minimal dashboard** for interaction

Think of Slice as **the brain and skeleton** of Liam's macro practice — everything consistent, logged, and monitored, but always with Liam as the decision-maker.

---

2. The Core Philosophy

Slice is built around a few core principles:

A. Human-in-the-loop macro

Slice does not explain "why markets moved."

It shows *what changed*, *how that affects your thesis*, and *whether your risk rails require action*.

B. Deterministic autonomy

All trading rules are explicit in your thesis:

- position bands
- max weight
- stops
- profit-taking
- timed adds/trims
- overall risk rails

The autonomy engine only executes what's pre-authorized.

There is no machine "intuition."

C. Slow macro, not reactive trading

Data checks run daily.

Expression performance is end-of-day.

Portfolio moves are measured and structured, not tick-by-tick.

D. Everyone speaks JSON

Theses and observations are structured objects living in the database.

This ensures clarity, consistency, and repeatability.

---

3. The Heart of Slice: ThesisJSON

A thesis is not a paragraph.

It is a **structured object** that the system can execute.

ThesisJSON includes:

1. Hypothesis

- what the user believes
- causal logic
- time horizon
- regime assumptions

2. Drivers

Macro variables that matter:

- real rates
- yield curve shape
- risk premium changes
- FX regime

Each driver includes:

- thresholds (warning/disconfirm)
- expected direction
- importance

3. Disconfirmers

Explicit statements of *what kills the idea.*

Examples:

- “If real yields > X for Y consecutive days”
- “If risk premium compresses despite tightening cycle”

This brings discipline and limits overconfidence.

4. Expressions

The actual way the thesis is expressed in the portfolio:

A thesis might have:

- a GLDM overweight
- long UUP
- SPY underweight
- A spread between TLT and SPTS

Each expression includes:

- target/min/max weights
- band-rebalancing rules
- stops
- profit-taking
- timed actions

5. Risk rails

Global constraints across the whole thesis:

- max exposure
- max concentration
- correlation limits
- autonomy permissions

6. Lifecycle

States of a thesis:

- DRAFT
- READY_REVIEW
- ACTIVE
- REVIEWED
- RETIRED

This ensures no half-baked thesis ever reaches autonomy.

---

4. Observations: The "Daily Mindstream"

Every thought, market impression, or intuition gets parsed into **ObservationJSON**.

Each includes:

- stance (confirming, disconfirming, mixed)
- linked drivers/disconfirmers
- certainty/tone
- market links
- possible impact on the thesis

They form the **history of your thinking**, and Slice tracks how observations age relative to reality.

---

5. Monitoring & Morning Briefing

Every morning Slice produces a structured report (via CLI or minimal web page):

1. Cross-asset surface

Levels and changes of the main tickers:

- SPY, QQQ, GLDM, UUP
- SGOV/SPTS, TLT/TBF
- FX

2. Headline summary

LLM extracts themes from raw news:

- no storytelling
- no narratives
- no explanations

Just:

- “Here’s what changed.”
- “Here’s the theme.”
- “Here are the tickers affected.”

3. Portfolio PnL & attribution

Linked back to expressions and theses.

4. Rates/FX radar

Yield curve shape, FX regime notes, and upcoming macro events.

---

6. Autonomy: Rule-Based Portfolio Management

The autonomy engine executes *only* the rules encoded in ThesisJSON.

It does four things:

1. **Band Rebalancing**

Keeps weights within min ↔ max, pulls toward target.

2. **Stops**

   Hard rules that override everything else.

   Highest execution priority.

3. **Profit-Taking**

   Triggers trims at predefined levels.

4. **Timed Actions**

   Scheduled adds/trims (e.g., "first Tuesday of the month if drivers supportive").

It runs **once per day** using EOD data.

All actions are logged in:

- Postgres (canonical record)
- human-readable log files (logs/autonomy/...)

---

7. Database & Storage

The system runs on **Postgres + pgvector**, giving one unified store for:

- price history
- macro series
- theses (JSONB + embeddings)
- observations (JSONB + embeddings)
- autonomy runs
- trades
- expression performance

FRED series are treated as "assets" so everything flows through the same pipeline.

Human-readable logs are stored in the repo so Liam can:

- see his thought evolution
- write papers
- reflect on process
- track changes day-to-day

---

8. LLM Usage

Slice uses LLMs in **5 very controlled places**:

1. **thesis_parser** — memo → structured ThesisJSON
2. **thesis_critic** — structural and causal critique
3. **observation_parser** — turns notes into structured objects
4. **headline_summarizer** — clusters news themes
5. **trade_proposer** — suggestions during thesis review session

Importantly:

LLMs **never execute trades**

LLMs **never override autonomy**

LLMs **cannot activate a thesis**

They are advisors and structure-builders, not agents.

---

9. Module Architecture (High-Level)

Everything lives under one clean Python package:

```
slice_app/
    db/
    schemas/
    thesis/
    autonomy/
    portfolio/
    quant/
    llm/
    briefing/
    scheduler/
    cli/
    api/
```

Core idea: layer separation

- **LLM modules** never touch execution or autonomy.
- **Autonomy** never calls LLMs.
- **CLI and web UI** only call service layers.
- **DB** is the backbone.

This keeps the system safe, predictable, and scalable.

---

10. Development Roadmap (Phase 2 Overview)

Sprint 1:

Repo + Postgres + migrations + initial CLI.

Sprint 2:

Ingestion for:

- Schwab API
- TwelveData price

Sprint 3:

Thesis lifecycle:

- schema
- parsing
- validation
- critique

Sprint 4:

Quant + expression performance + backtests.

Sprint 5

Autonomy in dry-run mode.

Sprint 6:

Morning Briefing builder.

By the end of Phase 2, Slice will be a functioning macro OS.

Conclusion

Slice is a rigorous, deeply structured, high-discipline macro-investing system built around:

- thoughtful thesis generation
- explicit rules
- clean autonomy
- daily process discipline
- minimal narratives
- maximum clarity