# DEEP AUDIT REPORT

## Polymarket Autonomous Trading Agent

Static Analysis & Code Review

Date: February 17, 2026

Prepared for: Alphha (pradeep.sriramadas@gmail.com)

Language: Rust | Framework: Tokio | Database: SQLite

**VERDICT: NOT READY FOR PRODUCTION**

7 critical bugs found. 1 prevents compilation. 4 affect financial calculations. 2 cause runtime panics.

# Executive Summary

This report presents a comprehensive static analysis of the Polymarket autonomous trading agent codebase. The audit covered all 40+ source files across 10 modules, examining financial math correctness, runtime safety, security, data integrity, and test coverage.

The analysis identified 30 distinct issues: 7 critical (including 1 compilation blocker), 9 high severity, 10 medium, and 4 low. The most consequential findings involve incorrect P&L calculations in the exit signal and resolution modules, a Cargo.toml edition that prevents compilation, and silent data corruption in the calibration system.

The agent has strong architectural foundations: proper use of Decimal for financial math, parameterized SQL queries, separation of concerns, and comprehensive risk management layers. However, several critical bugs in the financial calculation pipeline mean that **the agent will make incorrect trading decisions and report wrong P&L figures** in its current state.

## Issue Distribution

| CRITICAL | HIGH | MEDIUM | LOW |
|---|---|---|---|
| 7 | 9 | 10 | 4 |

# Detailed Findings

| ID | Severity | Module | Issue | Recommended Fix |
|---|---|---|---|---|
| **BLD-01** | CRITICAL | Cargo.toml | **Invalid Rust edition "2024"**<br>edition = "2024" does not exist. Valid editions: 2015, 2018, 2021. The project will not compile. | Change to edition = "2021" in Cargo.toml line 4. |
| **FIN-01** | CRITICAL | exit.rs | **NO side P&L calculation is wrong**<br>evaluate_exit() inverts entry_price for NO trades (1 - entry_price), but entry_price is already the NO token price. This double-inverts, causing wrong stop-loss triggers. A 50% loss shows as 67%. | For NO side: pnl_pct = (current_no_price - entry_price) / entry_price. Do not invert entry_price. |
| **FIN-02** | CRITICAL | metrics.rs | **total_pnl adds exposure instead of unrealized P&L**<br>total_pnl = realized_pnl + unrealized_exposure. But unrealized_exposure is notional (price * size), not profit. This makes total_pnl wildly inflated. | Compute unrealized_pnl = (current_price - entry_price) * size for each open trade, then add to realized_pnl. |
| **FIN-03** | CRITICAL | resolution.rs | **Silent default to NO-wins on parse failure**<br>If Gamma API returns missing/malformed outcome_prices, yes_price defaults to 0 and yes_won = false. All parse failures systematically credit NO winners. | Return Err() instead of defaulting. Log the malformed response and skip settlement. |
| **FIN-04** | CRITICAL | calibration.rs | **Corrupted fair_value silently defaults to 0.5**<br>If fair_value in DB is non-numeric, it defaults to 0.5, then forecast_correct may be wrong. This corrupts calibration over time. | Return Err() on parse failure. Skip corrupted rows in calibration computation. |
| **RUN-01** | CRITICAL | fair_value.rs | **String slice panic on multi-byte UTF-8**<br>truncate_json() uses &s[..max_len] which panics if max_len falls mid-character in UTF-8 (e.g., Chinese market questions). | Use s.chars().take(max_len).collect::<String>() instead of byte slicing. |
| **RUN-02** | CRITICAL | fair_value.rs | **f64 NaN/Infinity silently becomes probability 0.0**<br>Decimal::try_from(f64::NAN) fails, defaults to ZERO. Bounds check then passes (0 is in [0,1]). Agent treats NaN as 0% probability. | Check f64 is finite before conversion. Reject NaN/Infinity explicitly. |
| **TRD-01** | HIGH | edge.rs | **NO side trade_price may double-invert**<br>trade_price for NO = 1 - midpoint. If the order book is already for the NO token (not YES), this double-inverts. Depends on which token's book is passed. | Document clearly whether midpoint is always YES-side. Add assertion. |
| **TRD-02** | HIGH | self_funding.rs | **edge_justifies_cost ignores win probability** | Multiply by confidence: projected_profit = position_size * |

| | | | | |
|---|---|---|---|---|
| | | | Checks position * edge > api_cost, but edge is not expected value. Should be position * edge * confidence > cost to account for model uncertainty. | edge * confidence. |
| **TRD-03** | HIGH | portfolio.rs | **Constraint check uses full kelly_size, not adjusted**<br>check_constraints() validates against raw kelly_size, but adjust_size() may reduce it. Constraint check can reject trades that would actually fit. | Run adjust_size() before check_constraints(), or pass adjusted size. |
| **TRD-04** | HIGH | order.rs | **Token selection by array index, not by outcome**<br>YES = tokens[0], NO = tokens[1]. If Gamma API returns tokens in different order, wrong token is traded. | Match on token.outcome field instead of positional index. |
| **TRD-05** | HIGH | kelly.rs | **No guard for near-zero market_price**<br>market_price = 0.00001 passes the > 0 check but b = 1/0.00001 - 1 = 99999, producing unstable Kelly fractions. | Add minimum price threshold: if market_price < 0.01, return zero. |
| **TRD-06** | HIGH | lifecycle.rs | **Dead agent skips position cleanup**<br>When state = Dead, run_cycle() returns immediately. Open positions are never evaluated for exit or settlement. | Run resolution and exit evaluation before the Dead state early return. |
| **DAT-01** | HIGH | polymarket.rs | **Empty condition_id creates cache key collision**<br>If condition_id is None, it defaults to empty string. Two markets without IDs share the same valuation cache, returning wrong probabilities. | Skip markets where condition_id is empty. Never use empty string as cache key. |
| **DAT-02** | HIGH | fills.rs | **Corrupted trade data silently underreports exposure**<br>If entry_price or size fails to parse, the trade is skipped in exposure sum. This can allow over-leveraged positions. | Return Err() on parse failure instead of skipping. Or log warning and use conservative estimate. |
| **DAT-03** | HIGH | lifecycle.rs | **Uses notional exposure instead of P&L in survival check**<br>enhanced_survival_check receives unrealized_exposure (notional) but should use unrealized P&L. Overstates reserves. | Compute actual unrealized P&L by comparing entry vs current prices. |
| **SEC-01** | MEDIUM | dashboard.rs | **Dashboard has no authentication**<br>HTTP endpoints /api/trades, /api/cycles, /api/metrics are open to anyone. If bound to 0.0.0.0, full strategy is exposed. | Add API key auth or restrict to 127.0.0.1 only (default is already localhost). |
| **SEC-02** | MEDIUM | claude.rs | **API key could leak via tracing spans**<br>#[instrument] macro on complete() does not skip self. If DEBUG logging is enabled, the API key in self.api_key could appear in logs. | Add skip(self) or use a wrapper type with secure Debug impl. |
| **SEC-03** | MEDIUM | fair_value.rs | **Prompt injection defense is incomplete**<br>sanitize_market_question() strips | Add Unicode normalization. The system prompt defense-in-depth is good but add logging for suspicious |

| | | | | |
|---|---|---|---|---|
| | | | control chars and common patterns but doesn't block Unicode zero-width characters or logical injection. | patterns. |
| **COR-01** | MEDIUM | calibration.rs | **50% forecast always counted as correct**<br>fair_value == 0.5 is treated as correct regardless of outcome. Inflates calibration accuracy for uncertain predictions. | Treat 50% as incorrect (no directional prediction made). |
| **COR-02** | MEDIUM | resolution.rs | **No handling of fractional market resolutions**<br>Code assumes binary (0 or 1) outcomes. If Polymarket resolves fractionally (e.g., 0.6/0.4), P&L is wrong. | Support fractional resolution: pnl = (actual_price - entry_price) * size. |
| **COR-03** | MEDIUM | wallet.rs | **Decimal to f64 precision loss in cycles calc**<br>Converts Decimal -> String -> f64 -> u64 for cycles_remaining. Loses precision for large numbers. | Use Decimal::to_u64() or trunc() directly. |
| **COR-04** | MEDIUM | lifecycle.rs | **API cost check runs after the API call**<br>Bankroll check at line 342 happens inside loop after engine.evaluate() already spent money. | Move budget check before the evaluate() call. |
| **COR-05** | MEDIUM | health.rs | **Race condition in record_cycle**<br>record_cycle() spawns async task but doesn't await. Health check immediately after may return stale data. | Await the write or use atomic updates. |
| **COR-06** | MEDIUM | edge.rs | **Hardcoded confidence threshold 0.4 ignores config**<br>confidence < 0.4 filter is hardcoded, but config has high/low confidence edge thresholds. Inconsistent. | Read threshold from config or document the relationship. |
| **TST-01** | LOW | integration.rs | **Test assertion result discarded**<br>category_unknown_falls_back() uses matches!() without assert!(). Test always passes. | Change to assert!(matches!(cat, MarketCategory::Other(_))). |
| **TST-02** | LOW | exit.rs | **NO side test doesn't verify P&L magnitude**<br>Test only checks should_exit = true, not the actual pnl_pct value. Wrong magnitude goes undetected. | Add assert_eq! or assert! on signal.pnl_pct value. |
| **TST-03** | LOW | resolution.rs | **No test for malformed outcome_prices**<br>Missing tests for: invalid JSON, missing field, tie (0.5), fractional resolution. | Add tests for error paths and edge cases. |
| **TST-04** | LOW | order.rs | **No test for empty order books**<br>Missing test for midpoint fallback when bids/asks are empty. | Add test with empty bids and asks arrays. |

# Module Health Scorecard

| Module | Status | C | H | M | Key Issues |
|---|---|---|---|---|---|
| market/polymarket.rs | NEEDS WORK | 0 | 1 | 2 | Empty condition_id cache collision. Silent JSON parse failures. |
| valuation/fair_value.rs | CRITICAL | 2 | 1 | 1 | UTF-8 panic. NaN becomes 0%. Validation order wrong. |
| valuation/calibration.rs | NEEDS WORK | 1 | 0 | 1 | Corrupted data silently defaults to 0.5. 50% always correct. |
| risk/kelly.rs | NEEDS WORK | 0 | 1 | 1 | Near-zero price guard missing. Confidence unbounded. |
| risk/exit.rs | CRITICAL | 1 | 0 | 0 | NO side P&L formula is wrong. Stop-loss fires incorrectly. |
| risk/portfolio.rs | NEEDS WORK | 0 | 1 | 0 | Constraint check before size adjustment. |
| execution/order.rs | NEEDS WORK | 0 | 1 | 0 | Token selected by index, not outcome name. |
| execution/resolution.rs | CRITICAL | 1 | 0 | 1 | Parse failures default to NO wins. No fractional support. |
| execution/fills.rs | NEEDS WORK | 0 | 1 | 0 | Corrupted trades silently skipped in exposure. |
| agent/lifecycle.rs | NEEDS WORK | 0 | 2 | 2 | Dead state orphans positions. Exposure vs P&L confusion. |
| agent/self_funding.rs | NEEDS WORK | 0 | 1 | 0 | Edge justification ignores confidence. |
| monitoring/metrics.rs | CRITICAL | 1 | 0 | 0 | total_pnl adds notional exposure instead of P&L. |
| monitoring/dashboard.rs | OK | 0 | 0 | 1 | No auth, but defaults to localhost. |
| Cargo.toml | CRITICAL | 1 | 0 | 0 | edition = "2024" prevents compilation. |

# Recommended Fix Order

Fix these issues in order before running the agent, even in paper mode:

| # | Issue | Action | Effort | Why It Matters |
|---|---|---|---|---|
| 1 | BLD-01 | Change Cargo.toml edition to "2021" | 1 min | Nothing compiles without this |
| 2 | FIN-01 | Fix NO side P&L in exit.rs | 15 min | Wrong stop-loss triggers lose money |
| 3 | FIN-02 | Fix total_pnl in metrics.rs (use unrealized P&L, not exposure) | 30 min | All P&L reporting is wrong |
| 4 | FIN-03 | Return error on malformed resolution data instead of defaulting | 15 min | Wrong P&L attribution on parse errors |
| 5 | RUN-01 | Fix UTF-8 slice panic in fair_value.rs | 5 min | Runtime crash on non-ASCII market questions |
| 6 | RUN-02 | Reject NaN/Infinity before Decimal conversion | 10 min | Claude returning NaN treated as 0% probability |
| 7 | DAT-01 | Skip markets with empty condition_id | 5 min | Wrong valuations from cache collisions |
| 8 | TRD-04 | Match tokens by outcome name, not array index | 20 min | Trading wrong token if API order changes |
| 9 | TRD-06 | Run settlement/exit before Dead state early return | 15 min | Orphaned positions on agent death |
| 10 | TRD-02 | Multiply edge by confidence in self_funding check | 5 min | Agent trades when expected profit is negative |

# What the Codebase Does Well

1. Uses rust_decimal for ALL financial calculations. No f64 for money anywhere in the hot path.
2. All SQL queries use parameterized binds. Zero SQL injection risk.
3. Secrets loaded from environment variables only. Never stored in config files.
4. Kelly criterion implementation is mathematically correct for binary outcome markets.
5. Half-Kelly with confidence scaling prevents overbet. State-based multiplier reduces position sizing when low on funds.
6. Comprehensive risk layering: Kelly sizing, position limits, category diversification, stop-loss, daily budget cap.
7. Self-funding framework tracks whether edge justifies API cost. Agent stops trading when unprofitable.
8. Calibration system tracks Claude prediction accuracy over time and adjusts confidence discount.
9. Graceful shutdown with Ctrl+C via tokio::select!. No data loss on manual stop.
10. WAL journal mode on SQLite for crash safety. Concurrent reads supported.
11. Rate limiting with exponential backoff and configurable burst.
12. Data quality scoring filters out markets where Claude would be guessing without data.
13. Paper/Live/Backtest mode separation. Paper mode is functionally complete.
14. Dashboard defaults to 127.0.0.1 (localhost only).
15. Market question sanitization strips control characters and common prompt injection patterns.

# Conclusion

The Polymarket trading agent has a solid architectural foundation with proper financial math practices, good separation of concerns, and layered risk management. However, it has 7 critical bugs that must be fixed before any testing, even in paper mode.

The most urgent fix is trivial: changing edition = "2024" to "2021" in Cargo.toml. Without this, the project does not compile. The remaining critical fixes involve financial calculation errors that would cause the agent to make wrong trading decisions and report incorrect P&L.

After fixing the top 10 issues (estimated 2-3 hours of work), the agent should be ready for paper trading. Live trading requires additional work: Polymarket SDK wallet signing (EIP-712) is not yet implemented, and the API key exposed in git history must be rotated.

Estimated total effort to reach production-ready state: 2-3 hours for critical fixes, 1-2 days for all high/medium issues, plus the EIP-712 signing implementation for live mode.