

I'll conduct a comprehensive code review to identify potential clashes, errors, and document intended vs. actual functionality.

Critical Issues Found

1. Pool Size Mismatch Between Seed and Market Generator

Location: server/seed.ts vs server/market-generator-service.ts

- **Seed file** creates markets with pools: yesPool: "10000.00", noPool: "10000.00"
- **Market generator** creates markets with default pools: yesPool: "100.00", noPool: "100.00" (from schema defaults)

This creates inconsistent liquidity across the platform.

2. Missing Market Pool Initialization in Generator

Location: server/market-generator-service.ts

The InsertMarket objects created by the generator don't explicitly set yesPool and noPool, relying on schema defaults of "100.00" each. This is inconsistent with your seed data using "10000.00".

3. Price Calculation Inconsistency

Location: server/seed.ts line 94-95

```
const yesPrice = 0.40 + Math.random() * 0.20; // Random between 0.40 and 0.60
// But then sets: yesPrice: "0.5000", noPrice: "0.5000" (ignoring calculation)
```

The calculated yesPrice is never used - hardcoded to "0.5000".

4. Slippage Error Formatting Issue

Location: server/db-storage.ts line 456-460

The error message shows prices with 4 decimals but impact percentages - this could be confusing when percentages are very small.

5. Test Suite Using Old Pool Values

Location: server/tests/amm.test.ts

Tests expect pool depletion at largeAmount = 900 which was valid for \$200 total pools but won't trigger with \$20,000 pools.



Intended vs Actual Functionality

A. Market Pool Initialization

Intended:

- Consistent pool sizes across all markets
- \$20,000 total liquidity per market (\$10k YES + \$10k NO)
- Small trades (\$10) should have minimal price impact

Actual:

- **Seed markets:** \$20,000 total liquidity 
- **Generated markets:** \$200 total liquidity 
- Creates 100x difference in slippage between seeded vs scraped markets





B. Market Generation Diversity

Intended:

- Each KOL appears in exactly ONE market per generation cycle

- Minimum 2 head-to-head markets
- Maximum 30% SOL gain threshold markets
- Maximum 33% of any single type

Actual:





-  Correctly enforces one market per KOL
-  Generates minimum 2 head-to-head
-  Enforces type diversity limits
-  marketGenerationCount config parameter is deprecated but still exists in interface

C. AMM Price Impact & Slippage

Intended:

- Default 5% slippage tolerance
- Users can customize slippage
- Prevents trades that would move price >5% by default
- Price bounds: 0.01 - 0.99

Actual:





-  Slippage protection works correctly
-  Custom slippage supported
-  Price bounds enforced
-  Inconsistent pool sizes make slippage unpredictable across markets

D. Platform Fee System

Intended:

- 2% fee on buy orders
- Fee deducted before AMM calculation
- Fee tracked in platform_fees table

Actual:



-  Fee correctly deducted
-  Fee tracked in database
-  Net amount used for AMM calculation
-  Configurable via environment variable



E. Market Resolution

Intended:

- Auto-resolves markets after resolvesAt timestamp
- Settles all pending bets
- Updates user balances and stats
- Handles refunds for cancelled markets

Actual:

-  Auto-resolution runs every 5 minutes
-  Settlement logic works correctly





-  Refunds implemented
-  No metadata-based resolution yet (relies on manual resolution)

F. Concurrent Bet Handling

Intended:

- Row-level locking prevents race conditions
- Atomic transactions for all bet operations
- Consistent market state

Actual:





-  FOR UPDATE locks implemented
-  Full transaction isolation
-  Prevents double-spending
-  Prevents pool corruption

G. KOL Metrics Updates

Intended:

- Update every 30 minutes
- Store history snapshots
- Use Twitter API when available
- Fallback to mock data

Actual:





-  Auto-updates working
-  History tracked
-  Graceful Twitter API fallback
-  Failure tracking with max retry limit

H. Solana Integration

Intended:

- Users can deposit SOL to get platform balance
- Users can withdraw platform balance to SOL
- Deposit monitoring every 30 seconds
- Withdrawal processing

Actual:

-  Deposit address generation works
-  Monitoring service runs
-  Withdrawal processor runs
-  Requires SOLANA_HOT_WALLET_PRIVATE_KEY or generates temporary wallet

Recommended Fixes

server/market-generator-service.ts

Add explicit pool initialization to match seed.ts Add explicit pool initialization to match seed.ts Add pool initialization to all market generator methods (1/10)

Assistant

1 minute ago








Based on my comprehensive analysis of your codebase, here are advanced steps to make the system work better:

INTENDED vs ACTUAL FUNCTIONALITY

Intended Functionality:

1. **Market Creation:** Generate diverse markets from scraped KOL data with balanced pool sizes (\$20,000 total liquidity)
2. **AMM Trading:** Allow users to buy/sell shares with <5% price impact on small trades (\$10-50)
3. **Slippage Protection:** Reject trades exceeding 5% default slippage tolerance
4. **Market Resolution:** Automatically resolve markets based on scraped data comparisons
5. **Unique KOL Markets:** Each KOL appears in exactly ONE market per generation cycle
6. **Platform Fees:** Deduct 2% fee on buy orders to sustain platform

Actual Functionality (Issues Found):

1.  **Pool sizes increased** (you just applied this fix)
2.  **Tests still expect old pool sizes** (100.00) - will fail
3.  **Slippage validation triggers too aggressively** on legitimate small trades
4.  **No price impact caps** - trades can push prices to extremes (0.01-0.99)
5.  **Market metadata not used in resolution** - resolver doesn't check metadata table
6.  **No liquidity warnings** in UI when pools are low
7.  Platform fees working correctly (2% deduction)