#4156: Fix reordering issue in AzureOpenAiChatModel (Fixes #4155)

addressing the core ordering problem. However, the lack of test coverage is a significant quality concern.

⚠️ Technical Concerns

Outstanding Concerns:

- No automated test coverage for the race condition due to its non-deterministic nature
- Potential performance impact of switching from flatMap to flatMapSequential
- Lack of reproduction test case makes regression detection difficult
- Risk of similar ordering issues in other Spring AI chat model implementations
- Missing validation that the fix doesn't introduce latency in high-throughput scenarios

Complexity Indicators:

- Race condition is inherently difficult to test and reproduce consistently
- Reactive stream ordering semantics require deep understanding of Project Reactor
- Integration with Azure OpenAI streaming protocol adds complexity
- Tool execution flow adds additional async processing complexity
- Change affects critical chat response processing path

Recommendations:

- Add integration tests that can detect ordering violations even if not 100% reproducible
- Benchmark performance impact of flatMapSequential vs flatMap in realistic scenarios
- Review other Spring AI chat model implementations for similar flatMap ordering issues
- Consider adding documentation about streaming ordering guarantees in the API
- Implement monitoring or logging to detect ordering violations in production environments

Stakeholder Feedback:

🏗️ Solution Quality

Scope Analysis:

This is a highly focused change affecting only the AzureOpenAiChatModel class with 2 line modifications that replace flatMap() with flatMapSequential() in two locations within the reactive stream processing logic. The scope is appropriately narrow for addressing a specific reordering issue in chat completion streaming without affecting other Spring AI components or modules.

Architecture Impact:

- No changes to core Spring AI interfaces or public APIs - this is an internal implementation detail fix
- Aligns with reactive programming best practices by ensuring sequential processing where order matters
- Maintains existing Spring AI patterns while improving the reliability of streaming chat completions
- No impact on system modularity or dependency relationships - purely internal behavioral fix

Implementation Quality:

- Clean, targeted fix that addresses the root cause of reordering issues in reactive streams
- Uses appropriate Project Reactor operators (flatMapSequential) for maintaining order guarantees
- Changes are applied consistently across both affected stream operations in the same method context
- No new error handling needed as the operator change maintains the same error propagation semantics
- Integration with existing Spring Framework patterns remains intact
- The fix is located within a complex method (internalStream - 118 lines) but the change itself is surgical and appropriate
- No code duplication introduced and follows existing reactive stream patterns in the codebase

Testing Adequacy: