GenAI for ArbitrageGainer model feedback

GenAI feedback:

Feedback on your project submission

Strengths:

1. **Comprehensive Workflow Coverage**: Your workflows for core activities such as "Enter System," "Crypto Currencies List Loading," "Historical Arbitrage Calculation," "Real-time Trading," "P&L Calculation," and "Annualized Return Calculation" demonstrate a clear understanding of the system's processes. Each workflow is structured well, covering the necessary steps to achieve the objectives, such as validating user input, identifying cross-traded cryptocurrency pairs, and calculating arbitrage opportunities.
2. **Correct Use of Ubiquitous Language**: You have employed Domain-Driven Design principles effectively, particularly by using concise and clear event names such as `UserConsent`, `CrossTradedPairsStored`, `ArbitrageOpportunitiesPersisted`, and `OrderPersisted`. This aligns well with the need for clear and unambiguous terminology in the domain.
3. **Domain Event Modeling**: The events and commands in your workflows (e.g., `UserInputValid`, `CrossTradedPairsFound`, `ArbitrageOpportunitiesFound`) are modeled correctly and play their roles in advancing the workflows step by step. This event-driven approach adheres to the key principles of functional programming in practice.
4. **Handling of Trading Stop Conditions**: In workflows like "P&L Calculation" and "Real-time Trading," you have incorporated stop conditions, which are vital for ensuring that trading activities conclude correctly under specific conditions such as invalid orders or reaching the P&L target.

Weaknesses:

1. **Event and Command Definition Inconsistencies**: While events are used in the workflows, there are a few instances where the events or commands used (e.g., `InvalidUserInput`, `CrossTradedPairsStored`, `OrderPersisted`) lack explicit

definitions. These need to be clearly defined in your documentation to ensure consistency and clarity.

2. **Market Data Comparison**: Your workflows do not explicitly address how the comparison of market data for real-time trading is maintained to identify arbitrage opportunities. This is crucial, as the identification of arbitrage opportunities relies on the most recent data. You should include a mechanism for storing and updating the latest data.
3. **Workflow for Retrieving Order Status**: There is no specific step in the workflows related to retrieving the order status, which is an essential feature for ensuring that trades are executed properly. A clear mechanism for checking the status of placed orders and reacting to their outcomes (e.g., partial fills, cancellations) is missing.
4. **Handling Errors and Domain Errors**: While you have addressed certain errors like invalid user input, the workflows could benefit from a more detailed approach to error handling, particularly for potential domain errors in trading activities. For example, errors related to real-time data feeds, order validation, or P&L miscalculations should be explicitly modeled.

Areas of Improvement:

1. **Event Definitions**: Define all the events and commands used in the workflows explicitly. For example, for events like `InvalidUserInput` and `CrossTradedPairsStored`, provide their data structures and content, such as what data they carry and how they are processed.
2. **Real-Time Trading Data Management**: Add a step in your "Real-time Trading" workflow that clearly shows how the system compares and stores real-time market data for identifying arbitrage opportunities. This could involve creating a service that continuously retrieves and compares market data.
3. **Order Status Workflow**: Introduce a specific workflow for retrieving order status updates. This workflow should detail how the system tracks order fulfillment and manages partial or unfulfilled orders, aligning it with the requirements from the project specification.

By refining these areas, your project will better align with the requirements of the functional programming and domain-driven design principles. Keep up the good work!

Notes from feedback:

- GenAI had asked us to document some commands used. Since we are not implementing the application and are not creating documentation, we can ignore this feedback from AI
- Based on GenAI suggestion we had added an mechanism to retrieve real time data form exchanges and searches for arbitrage opportunities
- To keep track of past order info we have added an persistent DB
- We have added more error checks to make sure that the data we receive is as expected

What I have done:

- I made sure to clearly define the events and commands that interact with external systems. For example, I included `@CrossTradedPairsDBConnector` and `@HistoricalValuesFileConnector` to specify where the data is stored or retrieved. This ensures my workflows are consistent and easier to understand.
- I added clear stop conditions for workflows, like halting trading when the P&L target is reached or when an error occurs. This ensures my system can handle different outcomes effectively.