**1. Introduction**

This document outlines the system design for handling requests from the OPTIMUS chatbot via AMPS. The system will publish, subscribe, process, and respond to the chatbot queries from traders. The objective is to create a seamless, reliable, and efficient message-driven architecture for handling trader requests with minimal latency.

**2. High-Level Architecture**

The system follows an event-driven model, ensuring low-latency communication and fault tolerance. The architecture consists of interconnected components that facilitate seamless data flow between traders and backend processing services.

**Architecture Overview:**

1. **OPTIMUS Chatbot**: Serves as the interface for traders to send and receive messages.
2. **AMPS (Advanced Message Processing System)**: Acts as the messaging middleware for reliable request and response communication.
3. **Consumer Service**: Processes incoming trader requests and performs necessary computations.
4. **AMPS (Response Topic)**: Stores processed responses for the chatbot to retrieve.
5. **OPTIMUS Chatbot**: Retrieves the response from AMPS and displays it to traders.
6. **Trader**: Receives the processed response via the chatbot.

**3. Components**

**3.1 OPTIMUS Chatbot**

* The chatbot is responsible for receiving user queries and sending structured requests to AMPS.
* It interacts with traders using a user-friendly interface, possibly through a web-based application or mobile app.
* Once a trader submits a request, the chatbot formats the request in a structured format and publishes it to a designated AMPS topic.
* After the request is processed, the chatbot listens to the response topic and displays the results to the trader.

**3.2 AMPS (Advanced Message Processing System)**

* AMPS acts as the central messaging backbone of the system, ensuring real-time message delivery between different services.
* It provides reliable, low-latency message publishing and subscription mechanisms.
* The system will have two topics:
  + **Request Topic**: Where OPTIMUS publishes trader requests.
  + **Response Topic**: Where processed responses are sent back for OPTIMUS to consume.
* AMPS ensures message persistence, so in case of failures, messages are not lost.

**3.3 Consumer Service**

* The consumer service is responsible for processing trader requests received from AMPS.
* It subscribes to the **Request Topic**, listens for new messages, and processes them based on the requested functionality.
* The service performs necessary business logic computations and generates an appropriate response.
* The processed response is then published to the **Response Topic**, where OPTIMUS can retrieve and display it.
* The consumer service is implemented using **Java (Spring, Kafka, and AMPS SDK)** to ensure scalability and high performance.

**4. Workflow**

The following detailed workflow outlines how a trader’s request is processed:

1. **User Interaction** -> The trader sends a request using the OPTIMUS chatbot.
2. **Publishing to AMPS** -> OPTIMUS formats the request and publishes it to the **Request Topic** on AMPS.
3. **Listening & Processing** -> The consumer service subscribes to the **Request Topic**, fetches the message, and processes it.
4. **Sending Response** -> The processed response is published to the **Response Topic** in AMPS.
5. **Response Delivery** -> The chatbot subscribes to the **Response Topic**, retrieves the response, and formats it.
6. **Displaying Response** -> The chatbot presents the response to the trader.

1. Trader -> 2. OPTIMUS Chatbot -> 3. AMPS -> 4. Consumer Service -> 5. AMPS -> 6. OPTIMUS Chatbot -> 1. Trader