MVP Scope and Technical Analysis for Conversational AI Application

This document outlines the scope of a Minimum Viable Product (MVP) for a conversational AI application designed to assist financial advisors with meeting preparation. It also details the scope for analyzing technical options to determine the optimal architecture for transitioning the current microservices-based system into a multi-agent intelligent application. The document is divided into two main sections:

1. **MVP Scope Definition from User's Perspective**
2. **Scope for Analyzing Technical Options to Determine Optimal Architecture**

Section 1: MVP Scope Definition from User's Perspective

Objective

The MVP aims to reduce the time financial advisors spend on preparatory tasks—such as gathering client data, analyzing market trends, performing basic risk assessments, and generating reports—by providing a conversational AI application. This will enable advisors to focus more on direct client engagement.

Scope Overview

- **Core Features**:
  - Retrieve and display client portfolio summaries (e.g., total assets, recent transactions, performance).
  - Provide summaries of current market trends relevant to the client's investments.
  - Perform basic risk assessments (e.g., portfolio volatility, diversification analysis).
  - Generate reports combining client data, market trends, and risk assessments.
- **Conversational Interface**:
  - Advisors can interact with the AI using natural language.
  - Example queries:
    - "Show me the portfolio for John Doe."
    - "What are the market trends in the technology sector?"
    - "Perform a risk assessment for Jane Smith's portfolio."
    - "Prepare a report for my meeting with John Doe."
- **Integration**:
  - Connect to existing Spring Boot microservices for client data, market trends, and risk assessment logic.
  - Use AAD B2C OAuth2 for secure authentication.
- **User Experience**:
  - A web interface built with Angular for chat-based interaction.

    - Response times targeted at under 5 seconds.

Example Conversation

The following interaction illustrates how the MVP reduces preparatory time for advisors:

- Advisor: "Hi AI, I need to prepare for my meeting with John Doe."
- AI: "Sure, here's John Doe's portfolio summary:
    - Total Assets: $500,000
    - Recent Transactions: Sold 100 shares of AAPL, Bought 50 shares of MSFT
    - Performance: +5% over the last quarter
    - Current market trends relevant to his investments:
    - Technology sector up 3% this week, driven by AI advancements.
    - Energy sector showing volatility due to geopolitical tensions.
    - Would you like a basic risk assessment of his portfolio?"
- Advisor: "Yes, please."
- AI: "John Doe's portfolio has a moderate risk level:
    - Volatility: 12% (based on recent market fluctuations)
    - Diversification: 60% tech, 20% energy, 20% healthcare
    - Would you like me to generate a report?"
- Advisor: "Yes, generate a report."
- AI: "Report generated, including portfolio summary, market trends, and risk assessment. Download it here: [link to report]."

Value to Users

- **Time Savings**: Instantly retrieves client data, market insights, and risk assessments.
- **Efficiency**: Generates reports on demand, streamlining preparation.
- **Enhanced Client Focus**: Allows advisors to prioritize relationship-building and strategic discussions.

Section 2: Scope for Analyzing Technical Options to Determine Optimal Architecture

Objective

The MVP includes an analysis of different technical options to determine the optimal architecture for transitioning the existing microservices-based application into a multi-agent intelligent system. This system must integrate with APIs from various systems and support future autonomous capabilities.

Scope Overview

- **Current State**:
  - Microservices architecture built on Spring Boot.
  - Authentication via AAD B2C OAuth2.
- **Technical Options to Evaluate**:
  - **Spring AI with MCP**: Native Spring Boot integration, exposes services as MCP tools.
  - **LangChain/LangGraph (via LangChain4j)**: Java-compatible AI framework with advanced agent features.
  - **AutoGen**: Multi-agent framework with Java support.
  - **Semantic Kernel**: Microsoft's framework for LLM integration in Java applications.
- **Evaluation Criteria**:
  - Integration with Spring Boot
  - Scalability for multiple users
  - Multi-agent capabilities (e.g., autonomous task execution)
  - Development complexity and resource requirements
  - Future-proofing for advanced features
- **POC Scope**:
  - Develop a conversational interface using LangChain (Python) to compare with Spring AI.
  - Assess development speed, feature capabilities, and integration challenges.
- **Deliverables**:
  - A comparison report evaluating each option against the criteria.
  - A recommendation for the optimal architecture to support the transition.