Jira Epics and Stories based on the MVP scope defined. The scope focuses on a conversational AI application for financial advisors to reduce preparatory time by retrieving client data, analyzing market trends, performing risk assessments, and generating reports, with an Angular frontend, Spring Boot microservices integration, and a technical analysis component for future architecture.

**Epic 1: Conversational AI Application Development**

**Title**: Build Conversational AI Application for Advisor Meeting Preparation  
**Description**: Develop a conversational AI application that allows financial advisors to retrieve client portfolio data, market trends, perform risk assessments, and generate reports via a natural language interface, integrated with existing Spring Boot microservices and an Angular frontend.  
**Acceptance Criteria**:

* Advisors can interact with the AI using natural language queries.
* The application retrieves data from microservices and presents it accurately.
* The Angular frontend provides a functional chat interface.
* Responses are delivered within 5 seconds.

**Story 1.1: Client Portfolio Retrieval**

**Title**: Retrieve Client Portfolio Data via Conversational AI  
**Description**: Enable the AI to fetch and display client portfolio summaries (e.g., total assets, transactions, performance) from existing Spring Boot microservices when prompted by the advisor.  
**Acceptance Criteria**:

* AI responds to queries like "Show me the portfolio for John Doe" with accurate data.
* Data includes total assets, recent transactions, and performance metrics.
* Response is displayed in the Angular chat interface within 5 seconds.

**Story 1.2: Market Trends Summary**

**Title**: Provide Market Trends Summary via Conversational AI  
**Description**: Implement functionality for the AI to provide concise market trend summaries relevant to a client’s investments, sourced from static data or microservices.  
**Acceptance Criteria**:

* AI responds to queries like "What are the market trends in the technology sector?" with a brief summary.
* Summary reflects current trends (e.g., "Tech up 3% this week").
* Response is rendered in the Angular frontend within 5 seconds.

**Story 1.3: Basic Risk Assessment**

**Title**: Perform Basic Risk Assessment via Conversational AI  
**Description**: Add capability for the AI to perform a basic risk assessment of a client’s portfolio (e.g., volatility, diversification) when requested by the advisor.  
**Acceptance Criteria**:

* AI responds to "Perform a risk assessment for John Doe’s portfolio" with metrics like volatility (e.g., 12%) and diversification (e.g., 60% tech).
* Assessment is based on portfolio data from microservices or static logic.
* Results appear in the chat interface within 5 seconds.

**Story 1.4: Report Generation**

**Title**: Generate Preparation Report via Conversational AI  
**Description**: Enable the AI to compile a report including portfolio summary, market trends, and risk assessment, providing a downloadable link to the advisor.  
**Acceptance Criteria**:

* AI responds to "Generate a report for John Doe" with a downloadable PDF/HTML link.
* Report includes portfolio data, market trends, and risk assessment.
* Report is accessible via the Angular frontend within 5 seconds of request.

**Story 1.5: Angular Chat Interface**

**Title**: Develop Angular Frontend for Conversational Interaction  
**Description**: Build a simple Angular web frontend with a chat interface for advisors to input queries and view AI responses.  
**Acceptance Criteria**:

* Frontend displays a chat window for text input and AI responses.
* Supports natural language queries and renders formatted responses (e.g., lists, links).
* Interface is responsive and user-friendly on desktop browsers.

**Story 1.6: Microservices Integration**

**Title**: Integrate Conversational AI with Spring Boot Microservices  
**Description**: Connect the AI application to existing Spring Boot microservices for client data, market trends, and risk assessment, using AAD B2C OAuth2 for authentication.  
**Acceptance Criteria**:

* AI calls microservices securely with AAD B2C OAuth2 tokens.
* Data retrieval (portfolio, trends, risk) is accurate and consistent with microservice outputs.
* Integration supports concurrent advisor requests without errors.

**Epic 2: Technical Architecture Analysis**

**Title**: Analyze Technical Options for Multi-Agent Architecture Transition  
**Description**: Evaluate and compare technical frameworks (Spring AI with MCP, LangChain/LangGraph, AutoGen, Semantic Kernel) to determine the optimal architecture for transitioning the current Spring Boot microservices application into a multi-agent intelligent system capable of integrating with diverse APIs.  
**Acceptance Criteria**:

* A comparison report evaluates each option against defined criteria.
* A recommendation is provided for the optimal architecture.
* The analysis informs future development beyond the MVP.

**Story 2.1: Define Evaluation Criteria**

**Title**: Establish Criteria for Technical Option Analysis  
**Description**: Define clear criteria to evaluate technical options for transitioning to a multi-agent system (e.g., integration, scalability, complexity).  
**Acceptance Criteria**:

* Criteria include integration with Spring Boot, scalability, multi-agent capabilities, development complexity, and future-proofing.
* Criteria are documented and agreed upon by the team.
* Evaluation process is outlined for consistent application.

**Story 2.2: Spring AI with MCP Evaluation**

**Title**: Assess Spring AI with MCP as an Architecture Option  
**Description**: Analyze Spring AI with MCP for its suitability in integrating existing REST APIs and supporting multi-agent capabilities within the Spring Boot ecosystem.  
**Acceptance Criteria**:

* Assessment confirms seamless integration with Spring Boot microservices.
* Evaluates MCP’s ability to expose APIs as tools and support conversational AI.
* Results are documented with pros, cons, and feasibility notes.

**Story 2.3: LangChain POC Development**

**Title**: Build LangChain POC for Conversational Interface  
**Description**: Develop a proof-of-concept (POC) using LangChain (Python) to create a conversational interface that connects to Spring Boot REST APIs, comparing it to Spring AI.  
**Acceptance Criteria**:

* POC retrieves client portfolio data and market trends via REST API calls.
* Interface supports basic queries (e.g., "Show me John Doe’s portfolio").
* Development time and integration challenges are documented for comparison.

**Story 2.4: AutoGen and Semantic Kernel Evaluation**

**Title**: Evaluate AutoGen and Semantic Kernel as Options  
**Description**: Assess AutoGen and Semantic Kernel for their potential to support a multi-agent system integrated with Spring Boot microservices.  
**Acceptance Criteria**:

* Evaluation confirms Java compatibility and multi-agent features for each framework.
* Assesses integration feasibility with Spring Boot REST APIs.
* Results are documented with pros, cons, and feasibility notes.

**Story 2.5: Comparison Report and Recommendation**

**Title**: Compile Technical Options Comparison and Recommendation  
**Description**: Synthesize findings from Spring AI, LangChain POC, AutoGen, and Semantic Kernel evaluations into a report with a recommended architecture.  
**Acceptance Criteria**:

* Report compares options against evaluation criteria with clear pros and cons.
* Provides a single recommended architecture (e.g., Spring AI with MCP) with rationale.
* Delivered as a concise document for stakeholder review.

**Notes**

* **Timeline**: These Epics and Stories are scoped for a 2-3 week sprint by a small team (e.g., 4-5 members). Epic 1 focuses on immediate user value, while Epic 2 runs in parallel for technical analysis.
* **Team Split**: Most members (e.g., 3-4) work on Epic 1 (Java/Spring/Angular skills), with 1-2 handling Epic 2’s LangChain POC (Python skills).
* **Dependencies**: Story 1.6 (Microservices Integration) is foundational for Stories 1.1-1.4; Story 2.1 (Criteria) precedes 2.2-2.4.

This structure aligns with the MVP scope, providing actionable tasks with clear acceptance criteria to ensure delivery within the specified timeframe while addressing both user needs and future architecture planning. Let me know if you’d like to adjust priorities or add details!