AI Financial Advisor Project

Comprehensive Viva Questions & Model Answers

This document contains detailed explanations for 46 important questions covering Architecture, RAG, MCP Tools, Simulation, Memory Systems, Security, and AIML workflows in the AI Financial Advisor System.

1. Architecture & System Overview

1. What problem does your project solve?

It provides an AI-powered financial advisor capable of risk profiling, portfolio creation, Monte Carlo simulation, SEBI-compliant regulatory Q&A; using RAG, and report generation.

2. Why FastAPI?

FastAPI offers high performance, async support, type hinting, dependency injection, and automatic OpenAPI documentation—ideal for scalable AIML backends.

3. Why Streamlit?

Streamlit enables rapid prototyping of AI dashboards with minimal frontend effort, making it perfect for AIML trainees focusing on backend intelligence.

4. Why separate backend and frontend?

Clear separation of responsibilities ensures maintainability: backend handles intelligence, tools, memory, and APIs; frontend handles rendering and user interaction.

5. Why Azure OpenAI?

Azure provides enterprise-level GPT, embedding models, managed safety, reliability, and robust function-calling support essential for our tool-driven system.

2. MCP Tool Calling & LLM Integration

6. What is MCP-style tool calling?

A structured way for GPT to call backend functions. GPT identifies which tool to invoke and supplies JSON arguments; backend executes deterministic logic.

7. Why tool-calling over intent prediction?

Intent classifiers are rigid and brittle. Tool-calling allows GPT to dynamically orchestrate complex workflows by reading the tool schema.

8. Why not let GPT compute portfolio/simulation?

GPT is probabilistic and can hallucinate numbers. Financial computations must be deterministic and regulated, so backend tools perform all calculations.

9. How many tools were implemented?

Seven tools: risk_profile_tool, portfolio_tool, simulate_tool, rag_tool, nav_tool, currency_tool, set_investment_preferences.

10. Explain tool execution flow.

GPT → tool_call JSON → backend executes handler → result sent back → GPT generates final answer based on structured data.

3. RAG (Retrieval-Augmented Generation)

11. Why RAG?

To prevent hallucinations when answering SEBI regulations, mutual fund definitions, and tax rules. RAG ensures factual, document-grounded output.

12. Which retriever is used?

A FAISS-based dense retriever using Azure embeddings and L2 similarity search.

13. Describe your chunking method.

Fixed chunk size ~500 characters with 100-character overlap ensures contextual continuity and embedding precision.

14. Why FAISS?

Lightweight, offline-capable, highly optimized for vector search, and perfect for our project-scale dataset.

15. Explain the RAG workflow.

User query → Embed → FAISS search → Retrieve chunks → Feed chunks to GPT with strict instructions → GPT answers using retrieved text.

4. Memory System (Redis, SQLite, Semantic Cache)

16. Why use Redis?

Redis stores fast, session-specific memory enabling GPT to maintain stable context across multiple router calls.

17. What does Redis store?

User profile, risk category, SIP/tenure/goal data, last portfolio, last simulation, and optional chat summary.

18. Difference between Redis and SQLite?

Redis = short-term session memory; SQLite = permanent storage for users and chat logs.

19. What is semantic caching?

Embedding-based cache where semantically similar queries retrieve cached answers instantly, reducing GPT calls.

20. How does semantic caching work?

Embed query → FAISS cache search → if similarity > threshold, return cached answer; otherwise call GPT and store result.

5. Monte Carlo Simulation & Portfolio Logic

21. Why Monte Carlo simulation?

It models market uncertainty by simulating thousands of possible return paths instead of using deterministic formulas.

22. Key simulation parameters?

Expected return, volatility, asset weights, SIP/lumpsum, tenure, goal amount, and simulation count.

23. How is expected return (μ) computed?

Weighted average of asset-wise expected returns based on portfolio allocation.

24. How is SIP simulated monthly?

portfolio_value = portfolio_value * monthly_growth + monthly_sip, where monthly_growth is derived from annual sampled return.

25. How is goal-achievement probability computed?

Proportion of simulated final values exceeding the user's goal amount.

6. Risk Profiling & Portfolio Allocation

26. Why deterministic risk profiling?

Risk scoring must be transparent, explainable, and SEBI-compliant; ML models may behave unpredictably.

27. How is risk categorized?

Based on weighted scoring across age, liquidity, income stability, investment knowledge, and questionnaire.

28. Why deterministic portfolio allocation?

Allocation must be mathematically reliable, not LLM-generated, to prevent hallucination risk.

29. Example allocation mapping?

Conservative (20/60/10/10), Moderate (40/40/10/10), Aggressive (60/20/10/10).

7. Guardrails & Safety

30. What do input guardrails block?

Guaranteed returns, insider trading, tax evasion, market manipulation, or extreme-risk instructions.

31. What do output guardrails do?

Sanitize risky statements, enforce SEBI norms, and append mandatory disclaimers.

8. Authentication, Sessions & Security

32. Why not store plain passwords?

Security best practices require irreversible hashing via sha256_crypt.

33. Why both user_id and session_id?

user_id is long-term identity; session_id isolates advisory sessions, enabling multiple sessions per user.

34. What if session_id resets?

Redis loses context → portfolio and simulation fail; system asks user for missing information again.

9. PDF Generation & API Design

35. Why ReportLab?

It is backend-only, dependency-free, and ideal for structured PDF reports.

36. What does your PDF include?

User profile, risk summary, asset allocation, simulation results, and conversation summary.

37. Why modular routers?

Clear separation of logic—chat, risk, portfolio, simulation, auth, rag—improves maintainability.

38. How does frontend talk to backend?

Through APIClient using REST requests with session_id attached.

10. AIML-Specific & Advanced Questions

39. How would you deploy this system?

Containerize services, host FastAPI and Streamlit, use managed Redis, upgrade SQLite to PostgreSQL.

40. Why is this an AIML project?

Because it integrates LLM reasoning, RAG, semantic caching, tool orchestration, and probabilistic simulation.

41. How do you prevent GPT hallucination?

RAG grounding, deterministic tools, guardrails, and structured tool-calling constraints.

42. Can GPT modify portfolio directly?

No. GPT only calls portfolio_tool; backend defines deterministic logic.

43. What design choice are you proud of?

MCP tool orchestration—clean, scalable, and minimizes hallucinations.

44. What if a tool errors?

Backend catches it, returns an error object; GPT then adapts or asks for clarification.

45. Why not fine-tune GPT?

Fine-tuning is unnecessary—tools + RAG already give deterministic, accurate outputs.

46. What if Redis crashes?

Session memory is lost; advisor gracefully re-requests missing inputs and regenerates state.