Assignment 17 — Solutions

Project & Behavioural Questions

Part D — Interview Scenarios · 10 questions with answers

Q1. Walk me through a GenAI project you built end-to-end. What was your role and what was the business impact? ★ · Asked: TCS, Deloitte, Infosys, Google

Answer

How to answer (STAR + architecture, ~3 minutes):

- **Situation/Problem:** business context and why it mattered. "Fraud analysts spent ~20 minutes per case searching policy documents and past case notes."
- **Your role:** what *you* owned (design, RAG pipeline, evaluation, deployment) and team size.
- **Solution/architecture:** walk through data → ingestion/chunking → embeddings → vector DB → retrieval (hybrid + re-ranker) → LLM → guardrails → UI/API → monitoring. Name the tools and **why** you chose them.
- **Evaluation:** how you measured quality (golden set, RAGAS faithfulness, accuracy) and the numbers.
- **Result/impact:** quantified – "reduced investigation time from 20 to 6 minutes, used by 150 analysts, 85% answer acceptance".

Tips: prepare one project in depth with a diagram you can draw; be honest about what was a prototype vs production; expect follow-up questions on every component.

Q2. What was the hardest problem you faced in that project and how did you solve it? ★ · Asked: Google

Answer

Structure: challenge → why it was hard → options you considered → what you did → measurable result → lesson.

Example answer: "Our RAG bot gave wrong answers for questions about tables in scanned PDFs. Plain text extraction scrambled the rows, so retrieval found the right page but the LLM couldn't read the numbers. I compared three approaches – better OCR, a vision-LLM per page, and a table-aware parser – on 50 table questions. A layout-aware parser that converted tables to Markdown, kept each table as one chunk with an LLM-generated summary for retrieval, raised accuracy on table questions from 41% to 83%. Lesson: evaluate each component separately and fix the data pipeline before changing the model."

Pick a genuinely technical problem (data quality, hallucination, latency, cost, evaluation), show your reasoning and trade-offs, and quantify the outcome.

Q3. How did you evaluate your system and what metrics improved? · Asked: Google

Answer

Interviewers want to see a **disciplined evaluation process**, not just "it worked well":

- **Test set:** "We built a golden set of 150 real questions with expert-written answers and source documents, including out-of-scope questions."
- **Metrics by component:** retrieval – hit rate@5, context recall/precision; generation – faithfulness, answer relevancy/correctness (RAGAS + LLM-as-judge validated against human

labels); system – latency p95, cost per query.

- **Baselines and iterations:** "Baseline top-5 vector search: 68% hit rate → adding BM25 hybrid: 79% → adding a re-ranker: 88%; faithfulness improved from 0.81 to 0.93 after prompt changes."
- **Online metrics:** thumbs-up rate, escalation rate, adoption, time saved.
- **Continuous evaluation:** regression tests in CI on every prompt/model change; production failures added to the test set.

Q4. What would you do differently if you rebuilt it today?

Answer

Show **reflection and awareness of newer techniques**; be honest but positive. Example points:

- "Build the **evaluation set on day one** – we added it late and wasted time on changes we couldn't measure."
- "Use **hybrid search and a re-ranker** from the start instead of pure vector search."
- "Invest earlier in **document parsing** (tables, scanned PDFs), which caused most errors."
- "Add **observability/tracing** (Langfuse/LangSmith) before launch."
- "Use **LangGraph** for explicit control instead of a loosely controlled agent; add human-in-the-loop for risky actions."
- "Consider **smaller models + routing** and prompt caching to reduce cost."
- "Involve end users earlier to collect realistic queries."

Avoid blaming others; focus on what you learned.

Q5. How do you keep up with the fast-changing GenAI field?

Answer

Give **specific, credible habits** (and mention something recent you actually learned):

- **Papers & research:** arXiv / Hugging Face Daily Papers, Papers with Code; reading key papers (Attention Is All You Need, LoRA, RAG, ReAct, DPO).
- **Official docs and engineering blogs:** OpenAI, Anthropic, Google DeepMind, Meta AI, Hugging Face, LangChain/LlamaIndex blogs and release notes.
- **Newsletters/communities:** The Batch (DeepLearning.AI), Latent Space, Ahead of AI (Sebastian Raschka), Interconnects, r/LocalLLaMA, X/LinkedIn researchers, local meetups.
- **Courses:** DeepLearning.AI short courses, Hugging Face courses.
- **Hands-on:** build small projects to test new models/techniques (e.g. tried MCP servers, GraphRAG, a new open model), Kaggle, open-source contributions.
- **Teaching/sharing:** writing blog posts or teaching classes forces deep understanding.

Q6. Explain RAG (or a Transformer) to a non-technical manager in 2 minutes. ·

Asked: EY, Google

Answer

RAG explained simply (example):

"An AI model like ChatGPT is like a very smart new employee who has read a lot of the internet, but has **never seen our company's documents** and sometimes **guesses confidently** when unsure.

RAG gives this employee an **open-book exam**. When someone asks a question, the system first **searches our own documents** – policies, manuals, past cases – and picks the few most relevant pages. It then gives those pages to the AI and says: 'Answer **only** using these pages,

and tell us which page you used.'

Benefits: answers are based on our latest, approved information; every answer shows its source so people can verify it; when a policy changes we just update the documents – no expensive retraining; and sensitive documents can be restricted to the right people.

Limitations: it's only as good as the documents and the search – if the information isn't there, it should say 'I don't know'."

Tips: use an analogy, avoid jargon (embeddings, vectors), focus on business value, risks and cost; check for understanding.

(Transformer in one line for managers: "a model that reads all words of a sentence at once and learns which words to pay attention to, which is what makes modern AI understand context so well.")

Q7. Tell me about yourself (for a GenAI Engineer role). ★ · Asked: Infosys, Google

Answer

Use a **90-second, present → past → future** structure focused on GenAI:

- **Present:** current role and the GenAI system you own. "I'm a GenAI engineer at X, where I build RAG and agent-based assistants for fraud and operations teams…"
- **Highlights with numbers:** 1–2 projects, your contribution and impact (users, latency, accuracy, cost saved).
- **Skills:** LLMs, RAG, LangChain/LangGraph, evaluation, deployment (FastAPI, Docker, cloud), Python; domain expertise (e.g. fraud detection in banking).
- **Past (briefly):** how you moved from data science/ML into GenAI.
- **Future/why this role:** what you want to build next and why this company.

Keep it conversational, not a resume recital, and end on a hook the interviewer can ask about.

Q8. Tell me about a failure or production issue in your GenAI project and what you learned. ★ · Asked: Google

Answer

Use **STAR** and show ownership:

- **Situation:** "After launch, our policy assistant started giving outdated answers to some HR questions."
- **Task:** "I owned the RAG pipeline and had to find the cause quickly."
- **Action:** "Traces showed old and new versions of the same policy were both indexed; retrieval sometimes picked the old one. I added document versioning and an 'effective date' filter, removed superseded documents during ingestion, added a regression test set for policy-update questions, and an alert on ingestion failures."
- **Result:** "Outdated-answer complaints dropped to near zero; we now catch such issues before release."
- **Learning:** data lifecycle and evaluation matter as much as the model.

Choose a real failure, don't blame others, and focus on the fix and the prevention.

Q9. You disagree with a senior architect's technical decision (e.g. which framework or model to use). What do you do? · Asked: Infosys, Google

Answer

- **Understand first:** ask about their reasons and constraints (cost, team skills, compliance, timelines) – they may have context you don't.
- **Make it data-driven, not opinion-driven:** propose a small **proof-of-concept or benchmark** comparing both options on agreed criteria (accuracy on our eval set, latency, cost, maintainability).
- **Discuss privately and respectfully**, present results and trade-offs.
- **Disagree and commit:** once a decision is made, support it fully, document the risks, and set checkpoints to revisit if metrics show problems.

Example: "We disagreed on fine-tuning vs RAG; I ran a one-week comparison on 100 real questions – RAG matched quality at a fraction of the cost, and the architect agreed to go with it.

Q10. The client wants the solution in 2 weeks, but you estimate 6 weeks (or your team is lagging behind a deadline). What do you do? · Asked: Google, Deloitte

Answer

- **Don't just say yes or no** – understand what the client actually needs by the date (a demo? a pilot? production?).
- **Break the scope down** and propose options: an **MVP in 2 weeks** (core use case, limited documents/users, manual steps) and the full production version in 6 weeks; show what is cut and the risks of skipping (security, evaluation).
- Be transparent with data (estimates, dependencies) and agree on priorities in writing.
- If the team is lagging: identify blockers early, re-prioritize must-haves, remove low-value scope, ask for help/resources, communicate status and risks to stakeholders **early**, not on the deadline day.
- Never compromise on security/compliance to hit a date.