

1. MASTER AI / GENAI INTERVIEW QUESTION BANK

AI / ML / GenAI FUNDAMENTALS AI Fundamentals

2. What is Artificial Intelligence?
3. What is Machine Learning?
4. What is Deep Learning?
5. What is Generative AI?
6. What is the difference between AI, ML, Deep Learning, and Generative AI?
7. What is the difference between traditional ML and Generative AI?
8. What is an LLM?
9. How does an LLM work at a high level?
10. What is a token?
11. How does tokenization work?
12. What is a context window?
13. What happens when the input exceeds the context window?
14. What is temperature?
15. What is the difference between temperature and top-p?
16. What is hallucination?
17. Why do LLMs hallucinate?
18. How would you reduce hallucinations?

Transformers

19. Explain the Transformer architecture.
20. Why were Transformers introduced?
21. What is self-attention?
22. How does self-attention work?
23. What are Query, Key, and Value?
24. Why do we need Query, Key, and Value?
25. What is multi-head attention?
26. Why do we need multiple attention heads?
27. What is positional encoding?
28. Why is positional information required?
29. What is the difference between encoder and decoder?
30. How does a decoder-only model such as GPT work?
31. Why are Transformers better than RNNs/LSTMs for many NLP workloads?

EMBEDDINGS / VECTOR DATABASE

32. What are embeddings?
33. Why do we use embeddings?
34. How is text converted into an embedding?
35. What does an embedding vector represent?
36. What is cosine similarity?
37. Why is cosine similarity commonly used for semantic search?
38. What is Euclidean distance?
39. Cosine similarity vs Euclidean distance?
40. What is semantic similarity?
41. What is vector search?
42. What is top-K retrieval?
43. How do you select K?
44. What happens if K is too small?
45. What happens if K is too large?
46. What is a vector database?
47. Why do we need a vector database?
48. FAISS vs a production vector database?
49. How does FAISS perform similarity search?
50. What indexing strategies are available for vector search?
51. What is approximate nearest-neighbor search?
52. What is the difference between exact and approximate nearest-neighbor search?
53. OpenAI embeddings vs BGE embeddings vs Sentence Transformers?
54. When would you choose a local embedding model over an API-based embedding model?

RAG

55. What is RAG?
56. Explain RAG architecture end-to-end.
57. Why do we need RAG if we already have an LLM?
58. What problems does RAG solve?
59. What are the main components of a RAG pipeline?
60. Explain the complete document ingestion pipeline.
61. Explain the query pipeline in RAG.
62. What sort of data do you handle? Was it structured or unstructured data?
63. Does that unstructured data have images, tabular data, or proper text data?
64. How would you process structured data in RAG?
65. How would you process unstructured documents?

- 66. How would you handle PDFs?
- 67. How would you handle scanned PDFs?
- 68. How would you handle images inside PDFs?
- 69. How would you extract tables from PDFs?
- 70. How would you handle documents containing text, images, and tables?

Chunking

- 71. What is chunking?
- 72. Why do we need chunking?
- 73. How would you decide the chunk size?
- 74. What chunking strategy would you use?
- 75. What is chunk overlap?
- 76. Why do we need chunk overlap?
- 77. How would you decide the overlap?
- 78. What happens if chunks are too small?
- 79. What happens if chunks are too large?
- 80. Fixed-size chunking vs semantic chunking?
- 81. When would you use recursive character splitting?
- 82. When would you use semantic chunking?
- 83. How would you preserve document context while chunking?
- 84. How would you handle headings and sections during chunking?

Retrieval

- 85. How does retrieval work?
- 86. How do you retrieve the most relevant chunks?
- 87. How do you select top-K?
- 88. How would you improve retrieval accuracy?
- 89. What is hybrid search?
- 90. Why use hybrid search?
- 91. Vector search vs keyword search?
- 92. What is BM25?
- 93. How would you combine BM25 and vector search?
- 94. What is reranking?
- 95. Why would you use a reranker?
- 96. Where does reranking fit in the RAG pipeline?
- 97. How would you prevent irrelevant chunks from reaching the LLM?
- 98. What happens when no relevant document is found?
- 99. How would you handle ambiguous queries?

Updating / Ingestion

- 100. How would you update a PDF in a RAG system?
- 101. What happens to the old embeddings when a document is updated?
- 102. How would you detect that a document has changed?
- 103. How would you delete stale embeddings?
- 104. How would you maintain document versions?
- 105. How would you bulk upload 500 PDFs?
- 106. How would you design a scalable document-ingestion pipeline?
- 107. How would you handle failed documents during ingestion?
- 108. How would you retry failed ingestion jobs?
- 109. How would you prevent duplicate documents?
- 110. How would you process documents asynchronously?

111. RAG Evaluation

- 112. How do you evaluate a RAG system?
- 113. How do you measure retrieval quality?
- 114. How do you measure answer quality?
- 115. How would you detect hallucination?
- 116. How would you measure hallucination?
- 117. What metrics would you use for RAG evaluation?
- 118. What is context relevance?
- 119. What is context precision?
- 120. What is context recall?
- 121. What is faithfulness?
- 122. How would you create a RAG evaluation dataset?
- 123. How would you perform offline evaluation?
- 124. How would you perform online evaluation?
- 125. How would you compare two RAG implementations?

126. RAG PRODUCTION / SECURITY

- 127. How would you deploy RAG in production?
- 128. How would you scale a RAG system?
- 129. How would you handle thousands of concurrent users?
- 130. How would you reduce RAG latency?
- 131. Where does latency usually come from in a RAG pipeline?
- 132. How would you reduce embedding latency?
- 133. How would you reduce LLM latency?

- 134. How would you reduce token usage?
- 135. How would you reduce LLM cost?
- 136. How would you cache RAG responses?
- 137. What would you cache?
- 138. How would you monitor a production RAG system?
- 139. What metrics would you monitor?
- 140. How would you detect degradation in retrieval quality?
- 141. How would you handle API rate limits?
- 142. How would you handle model failures?
- 143. How would you implement model fallback?
- 144. How would you handle prompt injection?
- 145. What is indirect prompt injection?
- 146. How would you protect a RAG application against malicious documents?
- 147. How would you protect sensitive data?
- 148. How would you implement access control at document level?
- 149. How would you ensure one customer cannot retrieve another customer's documents?

150. AI AGENTS / AGENTIC AI

- 151. What is an AI agent?
- 152. What is the difference between an LLM application and an AI agent?
- 153. What makes an agent autonomous?
- 154. What are the main components of an AI agent?
- 155. What is tool calling?
- 156. How does an agent decide which tool to call?
- 157. What is an agent workflow?
- 158. Single-agent vs multi-agent architecture?
- 159. When would you choose multi-agent architecture?
- 160. When should you NOT use multi-agent architecture?
- 161. What are the disadvantages of multi-agent systems?
- 162. How do agents communicate with each other?
- 163. How do you maintain state between agents?
- 164. How do you validate agent outputs?
- 165. How do you handle an agent returning malformed output?
- 166. How do you handle an agent returning missing fields?
- 167. How do you handle an agent returning incorrect data?
- 168. How do you implement retries for agents?
- 169. How do you prevent infinite agent loops?

170. How do you implement timeouts?

171. How do you monitor agents?

172. How do you evaluate agents?

173. SUPERVISOR / MULTI-AGENT QUESTIONS

174. Explain a supervisor-based multi-agent architecture.

175. How does the supervisor decide which agent should handle a task?

176. What happens if the supervisor selects the wrong agent?

177. How would you validate the supervisor's decision?

178. How would you force the supervisor to return structured output?

179. How would you include supporting evidence in the supervisor response?

180. Actual interviewer follow-up

181. "Walk me through the exact text format you'd use to force that supervisor workflow to return a single structured decision plus supporting evidence, and how you'd handle cases where one agent returns malformed or missing fields."

182. How would you validate outputs of agents?

183. You spoke about the DP val framework and all that. How would you apply that to an actual use case?

184. "Let's assume this use case, right? I mean, we are extracting the number of defects, or extracting the defects..."

185. How would you validate the extracted defect information?

186. What happens if one agent gives a malformed response?

187. What happens if one agent returns missing fields?

188. How would the supervisor recover from an agent failure?

189. AGENT WORKFLOW STATE / PAUSE & RESUME

190. How would you persist agent state?

191. How would you resume an agent workflow after failure?

192. How would you make a multi-step workflow fault tolerant?

193. How would you implement checkpoints?

194. How would you make an agent workflow idempotent?

195. How would you handle retries without duplicating actions?

196. How would you recover from a partially completed workflow?

197. How would you persist intermediate results?

198. Actual interviewer question

199. "How would you build an e-commerce returns assistant using a generative AI framework so that a multi-step run can be paused and resumed across retries without losing progress?"
200. How would you design the state object for this workflow?
201. Where would you persist the state?
202. What happens if the application crashes in the middle of the workflow?
203. How would you resume from the last successful step?
204. How would you prevent duplicate tool calls after resumption?

205. E-COMMERCE RETURNS AGENT

206. Design an AI-powered e-commerce returns assistant.
207. What agents would you create for the returns assistant?
208. What tools would the agents need?
209. How would the supervisor route the request?
210. How would you validate whether an item is eligible for return?
211. How would you retrieve the customer's order?
212. How would you handle an invalid order ID?
213. How would you handle a return-policy exception?
214. How would you handle human escalation?
215. How would you persist workflow state?
216. How would you handle failures?
217. How would you resume a failed workflow?
218. How would you ensure the customer is not charged/refunded twice?
219. How would you audit the agent's decisions?

220. PYTHON

221. Explain Python's major data structures.
222. List vs tuple?
223. Set vs list?
224. Dictionary vs list?
225. Mutable vs immutable objects?
226. What is shallow copy?
227. What is deep copy?
228. What are Python generators?
229. What are iterators?

- 230. Iterator vs generator?
- 231. What are decorators?
- 232. What are lambda functions?
- 233. What are list comprehensions?
- 234. What are *args and **kwargs?
- 235. Why do we use *args and **kwargs?
- 236. Why not simply use normal arguments?
- 237. What is the difference between args and kwargs?
- 238. What is exception handling?
- 239. try/except/finally?
- 240. How does Python memory management work?
- 241. What is garbage collection?
- 242. What is a Python virtual environment?
- 243. What is OOP?
- 244. Explain inheritance.
- 245. Explain polymorphism.
- 246. Explain encapsulation.
- 247. Explain abstraction.
- 248. Class method vs static method vs instance method?
- 249. What is method overriding?
- 250. What is method overloading in Python?
- 251. How does Python handle concurrency?
- 252. Multithreading vs multiprocessing?
- 253. Async vs synchronous programming?
- 254. When would you use asyncio?
- 255. What is the GIL?
- 256. How would you optimize Python code?

257. PYTHON CODING / OUTPUT QUESTIONS

258. What will be the output of a function using a mutable default argument?

259. Example discussed:

```
260.     def add_item(item, items=[]):
           items.append(item)
           return items

print(add_item(1))
print(add_item(2))
print(add_item(3))
```

- 261. Why does the mutable default argument retain values between calls?
- 262. How would you fix the mutable default argument problem?
- 263. Find the missing number from an array containing numbers from 1 to N.
- 264. Solve the missing-number problem using the sum approach.
- 265. Solve the missing-number problem using XOR.
- 266. What is the time complexity of your solution?
- 267. What is the space complexity?
- 268. How would you optimize the solution?

269. LANGCHAIN / GENERATIVE AI FRAMEWORKS

- 270. What is LangChain?
- 271. Why would you use LangChain?
- 272. What are chains?
- 273. What are agents?
- 274. What are tools?
- 275. What is LangGraph?
- 276. LangChain vs LangGraph?
- 277. When would you use LangGraph?
- 278. How do you maintain state in LangGraph?
- 279. How do you implement retries?
- 280. How do you implement checkpoints?
- 281. How do you implement human-in-the-loop?
- 282. How would you build a multi-agent workflow using a GenAI framework?
- 283. How would you validate structured outputs?
- 284. How would you handle malformed structured output?

285. PROMPT ENGINEERING

- 286. What is prompt engineering?
- 287. Zero-shot vs few-shot prompting?
- 288. What is chain-of-thought prompting?
- 289. What is structured output?
- 290. Why use JSON output?
- 291. How do you enforce a JSON schema?
- 292. What happens if an LLM does not follow the schema?
- 293. How would you retry malformed JSON?
- 294. How would you constrain an LLM's response?

- 295. How would you design prompts for agents?
- 296. How would you prevent prompt injection?
- 297. How would you separate system instructions from user input?
- 298. How would you version prompts?

299. LLM MODEL SELECTION

- 300. How would you select an LLM for a production application?
- 301. OpenAI vs Gemini vs open-source models?
- 302. Hosted model vs local model?
- 303. When would you use Ollama?
- 304. When would you use an open-source LLM?
- 305. How do you compare LLM models?
- 306. What metrics would you use?
- 307. How would you balance quality, cost, and latency?
- 308. How would you implement model fallback?
- 309. How would you route different queries to different models?
- 310. How would you handle model version changes?

311. DATABASE / BACKEND

- 312. SQL vs NoSQL?
- 313. When would you use PostgreSQL?
- 314. When would you use MongoDB?
- 315. How would you store conversation history?
- 316. How would you store embeddings?
- 317. How would you store metadata?
- 318. How would you design the schema for a RAG application?
- 319. How would you implement multi-tenancy?
- 320. How would you implement authentication?
- 321. How would you implement authorization?
- 322. How would you secure API endpoints?

323. API / BACKEND / FASTAPI

- 324. Why use FastAPI?
- 325. FastAPI vs Flask?

- 326. How do you build an API for an LLM application?
- 327. How would you stream an LLM response?
- 328. What is asynchronous API processing?
- 329. How would you handle concurrent requests?
- 330. How would you implement request validation?
- 331. How would you handle API errors?
- 332. How would you implement authentication?
- 333. How would you monitor API performance?
- 334. How would you scale a FastAPI service?

335. SYSTEM DESIGN

- 336. Design a production-grade RAG system.
- 337. Design a scalable chatbot.
- 338. Design an enterprise document Q&A system.
- 339. Design a multi-agent AI system.
- 340. Design an AI customer-support system.
- 341. Design an e-commerce returns assistant.
- 342. Design an AI document-processing pipeline.
- 343. Design an AI interview assistant.
- 344. How would you make the system highly available?
- 345. How would you make the system fault tolerant?
- 346. How would you handle millions of documents?
- 347. How would you handle millions of users?
- 348. How would you design caching?
- 349. How would you design asynchronous processing?
- 350. How would you design monitoring and observability?
- 351. How would you design authentication and authorization?

352. PRODUCTION / MLOPS / OBSERVABILITY

- 353. How would you deploy an AI application?
- 354. How would you containerize it?
- 355. Why Docker?
- 356. How would you deploy using Kubernetes?
- 357. How would you scale the application?
- 358. How would you implement CI/CD?
- 359. How would you monitor LLM latency?

- 360. How would you monitor token consumption?
- 361. How would you monitor cost?
- 362. How would you monitor hallucination?
- 363. How would you monitor retrieval quality?
- 364. How would you trace an agent workflow?
- 365. How would you debug a production LLM application?
- 366. How would you handle failures in production?
- 367. How would you implement retries?
- 368. How would you implement circuit breakers?
- 369. How would you implement fallbacks?

370. PROJECT / EXPERIENCE QUESTIONS

- 371. Tell me about yourself.
- 372. Walk me through your current project.
- 373. Explain your AI/GenAI project.
- 374. Explain your RAG project.
- 375. Explain the architecture of your project.
- 376. What was your role in the project?
- 377. What challenges did you face?
- 378. How did you solve those challenges?
- 379. Why did you choose this technology?
- 380. Why did you choose this architecture?
- 381. What alternatives did you consider?
- 382. What would you change if you rebuilt the project?
- 383. What was the biggest technical challenge?
- 384. How did you validate your solution?
- 385. How did you measure success?
- 386. How did you handle production issues?
- 387. How did you handle failures?
- 388. How did you optimize performance?
- 389. How did you reduce cost?
- 390. How did you deploy the project?
- 391. How did you monitor the project?

392. PERSONAL PROJECT QUESTIONS

- 393. Tell me about your personal projects.

- 394. Which personal project are you most proud of?
- 395. Why did you build it?
- 396. What problem does it solve?
- 397. What technology stack did you use?
- 398. Why did you choose that stack?
- 399. Did you build it yourself?
- 400. What part did you personally implement?
- 401. Is the project deployed?
- 402. Is it available on the Play Store?
- 403. How many users does it have?
- 404. How did you monetize it?
- 405. What did you learn from the project?
- 406. What would you improve?
- 407. What was the hardest part?
- 408. "Personal projects, right? Are they on Play Store?"

409. FOLLOW-UP / CHALLENGE QUESTIONS

- 410. Why did you choose this approach?
- 411. What if your assumption is wrong?
- 412. What happens if the model fails?
- 413. What happens if retrieval fails?
- 414. What happens if the database is unavailable?
- 415. What happens if the API times out?
- 416. What happens if the agent gives an invalid response?
- 417. How would you recover?
- 418. How would you test it?
- 419. How would you scale it?
- 420. How would you reduce latency?
- 421. How would you reduce cost?
- 422. How would you monitor it?
- 423. How would you prove that your approach works?
- 424. What are the trade-offs?
- 425. What alternative architecture could you use?
- 426. Why is your solution better than the alternative?

427. INTERVIEW META / BEHAVIORAL

- 428. Tell me about yourself.
- 429. Walk me through your resume.
- 430. Why are you looking for a change?
- 431. Why do you want this role?
- 432. Why should we hire you?
- 433. What are your strengths?
- 434. What are your weaknesses?
- 435. Tell me about a difficult technical problem you solved.
- 436. Tell me about a failure.
- 437. Tell me about a conflict with a teammate.
- 438. How do you handle pressure?
- 439. How do you prioritize tasks?
- 440. How do you work with cross-functional teams?
- 441. Where do you see yourself in the next few years?
- 442. Why are you moving from QA/automation toward AI/GenAI?
- 443. Why should we consider you for an AI Engineer role?

444. HIGH-PRIORITY QUESTIONS FOR YOUR AI ENGINEER INTERVIEWS

- 445. These are the questions that repeatedly matter most for the interviews I've been preparing for:
- 446. Explain your RAG architecture end-to-end.
- 447. How do you choose chunk size and overlap?
- 448. How do you improve retrieval quality?
- 449. Explain hybrid search.
- 450. How do you evaluate RAG?
- 451. How do you detect hallucination?
- 452. How do you handle prompt injection?
- 453. How do you scale RAG?
- 454. How do you reduce latency and cost?
- 455. Explain single-agent vs multi-agent.
- 456. Why use a supervisor?
- 457. How do you validate agent outputs?
- 458. How do you handle malformed agent responses?

- 459. How do you persist agent state?
- 460. How do you pause and resume an agent workflow?
- 461. How do you implement retries safely?
- 462. How do you force structured output?
- 463. How do you design an e-commerce returns agent?
- 464. How do you monitor an agentic application?
- 465. How do you choose an LLM?
- 466. Why LangGraph/LangChain?
- 467. How would you deploy your GenAI system in production?
- 468. How would you secure it?
- 469. How would you scale it?

- 470. Approximate consolidated question bank: **490 questions / question variants**
- 471. The list intentionally includes:
- 472. Original questions
- 473. Distinct follow-ups
- 474. Architecture questions
- 475. "What if..." challenge questions
- 476. Coding questions
- 477. Project questions
- 478. RAG questions
- 479. Agent questions
- 480. Production questions
- 481. QA/automation questions
- 482. Behavioral questions
- 483. Questions recovered nearly verbatim from prior interview conversations
- 484. Duplicates that were clearly asking the same thing have been consolidated where appropriate.