✅

1. FastAPI (for Agentic Workspace)

You must know:

- How FastAPI handles requests internally (ASGI, Starlette)
- Dependency injection
- Async vs sync endpoints
- Handling file uploads & streaming (SSE — which you used!)
- Background tasks
- How middlewares work
- Pydantic models + validation
- Rate limiting patterns (even high-level)

Interview-level questions:

- How does FastAPI achieve high performance?
- When to use async vs sync DB calls?
- How would you scale a FastAPI app?
- How does SSE work in FastAPI?

Since your RAG system uses streaming, they WILL ask about SSE.

✅

2. PostgreSQL + pgvector

This is **very important** because pgvector + hybrid search is rare for freshers; it gives you a huge advantage.

You must know:

- How vectors are stored in Postgres
- What an index (IVFFlat/HNSW) is
- How approximate vector search works
- Why hybrid search (BM25 + vector) is needed
- How you did cross-encoder re-ranking

Interview-level questions:

- Why use pgvector instead of ElasticSearch?
- Difference between cosine similarity and dot product?
- How is a vector index built?
- How do you prevent slow vector queries?

✅

3. Redis (for URL Shortener)

Redis is *the most important* technology for backend interviews.

You must know:

- In-memory storage architecture
- Data types (string, hash, list, set, sorted set)
- Expiry mechanism
- Eviction policies
- Redis persistence (AOF, RDB)
- Redis atomic counters → **You used this**
- Pipelining & transactions
- Caching strategies (write-through, write-back, write-around)

Interview-level questions:

- How does Redis guarantee atomicity for counters?
- What happens if Redis crashes?
- How do you prevent cache stampede?
- Why Redis over DB indexing?

Since your project explicitly says “85% load reduction” — expect deep questions.

✅

4. Spring Boot (for LMS)

This is another high-value topic.

You must know:

- Bean lifecycle, dependency injection (IoC)
- REST controller flow
- Filters vs Interceptors
- Spring Security → especially JWT auth
- JPA/Hibernate basics
- Transaction isolation levels
- Pagination, batching
- How to design scalable APIs

Interview-level questions:

- How does @Transactional work internally?
- How does JWT authentication work end-to-end?
- What is the difference between filter and interceptor?
- How to scale a Spring Boot app horizontally?

✅

5. Docker + AWS (you used EC2, S3, RDS)

You don't need deep DevOps knowledge, but you **must** know:

For Docker:

- Dockerfile basics
- Container lifecycle
- Image layers
- Why containerization makes deployment easier

For AWS:

- EC2 → Deploying backend
- RDS → Performance + scaling basics
- S3 → File storage + presigned URLs
- Security groups
- Load balancers (high-level)

Interview questions:

- What is the difference between VM and container?
- How would you deploy your LMS system on AWS?
- Why use S3 instead of storing files in database?

🎯

How Deep Should You Study Each Tech?

Not too deep — no need to know kernel-level internals.

You need:

Enough depth to defend design choices + answer “why” questions.

For example:

Skill	**Level needed**
FastAPI	Interview-ready + architecture awareness
pgvector	Conceptual + indexing strategies
Redis	Medium-deep (highly asked in interviews)
Spring Boot	Medium-deep (company dependent)
JWT, Auth	Deep (critical in backend interviews)
AWS, Docker	Medium (basic deploy knowledge)

🎯

The Most Important Areas (Ranked for Interviews)

If you have limited time, focus like this:

1️⃣ **Caching + Redis internals**

2️⃣ **Authentication + JWT + RBAC**

3️⃣ **System design for your 3 projects**

4️⃣ **Postgres + indexing (incl. pgvector)**

5️⃣ **FastAPI internals (async, DI, SSE)**

6️⃣ **Spring Boot internals**

7️⃣ **AWS + Docker basics**

If you master the above → you will 100% crack backend roles.

Must Know AI/ML interview questions -

1. RAG (Retrieval-Augmented Generation) — GUARANTEED Questions

Since your project uses hybrid search + pgvector + agents, expect heavy RAG questions.

Mandatory concepts you must be able to explain:

🔹 **What is RAG? Why use it instead of directly prompting the LLM?**

Explain: hallucination reduction, grounding, domain adaptation.

🔹 **Retrieval pipeline**

Be able to describe:

Document ingestion → Chunking → Embedding → Indexing → Retrieval → Re-ranking → Prompt construction → LLM response

🔹 **Chunking strategies**

They may ask:

- What chunk size did you use?
- Why overlapping?
- How chunking affects retrieval quality?

🔹 **Embeddings**

- What embedding model did you use? Why?
- Cosine similarity vs dot product
- Dimensionality trade-offs
- How embedding quality affects retrieval

✅

2. Vector Databases + pgvector — HIGH PROBABILITY

Because your backend uses **PostgreSQL + pgvector**, interviewers will dig here.

They may ask:

- How are vectors stored in pgvector?
- What is IVFFlat or HNSW?
- How approximate nearest neighbor (ANN) search works?
- How to choose nlist, nprobe?
- How do indexes speed up vector search?
- When to use vector search vs keyword search (BM25)?
- Why hybrid search is needed?

✅

3. Hybrid Search (BM25 + Vector Search) — VERY HIGH PROBABILITY

Your resume explicitly mentions:

“hybrid search (vector + BM25) with cross-encoder re-ranking.”

Expect questions like:

🔹 Why use hybrid search?

Explain:

- BM25 handles exact lexical matches
- Vector search handles semantics
- Combining improves both relevance + recall

🔹 How do you combine scores from BM25 + vector similarity?

Talk about:

- Weighted sum
- Reciprocal rank fusion
- Normalization

🔹 What is cross-encoder re-ranking?

Explain:

- Bi-encoder → fast, approximate
- Cross-encoder → slow but highly accurate (token-level attention)
- Used only on the top-k retrieved results

✅

4. LangChain Agents — MEDIUM PROBABILITY

You implemented:

- AI agents for email drafting, Jira ticket creation, report generation

They may ask:

- What is an agent in the LangChain framework?
- How is an agent different from a chain?

- What tools did your agent use?
- How do you prevent infinite loops?
- How do you ensure safe tool invocation?

Bonus:

- ReAct prompting
- Function calling vs tools
- Agent memory

✅

5. LLM Internals & Prompt Engineering — MEDIUM PROBABILITY

Not very deep — just conceptual.

You must know:

- What is a transformer? (High-level)
- What is attention? (Explain simply)
- Differences between:
  - **Prompting**
  - **Fine-tuning**
  - **RAG**
  - **Adapters (LoRA)**

Expect questions like:

- Why did you choose RAG over fine-tuning?
- How do you reduce hallucinations?
- How do you design prompts for your system?

✅

6. Evaluation of RAG Systems — INTERVIEWERS LOVE THIS

Most candidates have no idea about evaluation.

They may ask:

- How do you measure retrieval quality?
    - Recall@k
    - Precision
    - Hit rate
- How do you test RAG correctness?
- How do you evaluate hallucinations?
- How do you benchmark latency?

✅

7. Scalability & Architecture — HIGH PROBABILITY

Your RAG system includes:

- SSE streaming
- Multi-tenant design
- Hybrid search

Expect:

- How did you scale embedding generation?
- How do you handle large documents?
- How do you store metadata for chunks?
- How does SSE work?
- How would you scale pgvector?
- How do you handle concurrent user queries?

✅

8. Data Pipeline Questions

Since you built ingestion:

Questions they can ask:

- How do you parse PDFs?
- How do you handle tables, images?
- What happens if a huge file is uploaded?
- How do you ensure async job execution?
- Did you do offline indexing or on-the-fly?

✅

9. Security + Multi-Tenancy (Rare but valuable)

Your resume says:

> "multi-tenant RAG platform with JWT auth and role-based access."

Expect:

- How did you separate embeddings per tenant?
- How do you prevent data leakage?
- How do you handle access control for documents?
- Why JWT and not session-based auth?

❤️

10. Behavioral Questions Specific to AI Projects

Interviewers will ask:

“What was the most difficult challenge in building this AI system?”

You should talk about:

- Retrieval quality
- Latency
- Large file processing
- Accuracy tuning
- Prompt failure cases

“What is one thing you would improve in your RAG system?”

Say:

- Structured retrieval
- Add reranking
- Add caching for embeddings
- Move to dedicated vector DB (Weaviate, Pinecone)

“What failures did you face and how did you solve them?”

Talk about:

- Wrong retrieval
- Multi-tenant isolation
- Query latency
- Long document chunking issues

🎯 SUMMARY: What AI/ML Topics You MUST Prepare (Your Checklist)

🔹 MUST KNOW (Top priority):

- RAG pipeline end-to-end
- Chunking & embeddings
- pgvector indexing
- Hybrid search
- Cross-encoder re-ranking
- LangChain Agents
- Retrieval evaluation metrics
- Scaling/latency optimization

🔹 GOOD TO KNOW (Extra points):

- Transformers high-level
- Attention mechanism
- Why RAG > fine-tuning
- Safety, hallucination reduction

🔹 DO NOT NEED:

❌ Backpropagation

❌ CNNs, RNNs

❌ Deep learning math

❌ ML algorithms like SVM, random forest

❌ Statistical ML

Because your project is **LLM application architecture**, not ML research.