PERSONALIZED PREPARATION HANDBOOK

Global Remote AI Engineering Interview Preparation

A job-description-driven syllabus for Senior Applied AI, AI Platform, Forward Deployed and AI Architecture roles.

PREPARED FOR	TARGET ROLES
Purnendu Das Senior Generative AI Developer	Senior / Lead / Staff AI Engineer AI Architect / Forward Deployed Engineer

Core strategy: deepen production search and RAG, rigorous evaluation, enterprise integration, reliability and customer-facing architecture. Do not spend your main preparation time on foundation-model training mathematics.

Based on current official roles from Qdrant, Automattic, Deel, Remote, Sourcegraph, Canonical and Supabase.

01

The preparation strategy

Turn your existing GenAI experience into interview-ready evidence.

You already have strong exposure to Python, FastAPI, RAG, LangGraph, AWS/GCP, document AI, guardrails and LLM gateways. The plan below therefore emphasizes the areas that most improve your odds in senior global-remote interviews.

AREA	EFFORT	DESIRED INTERVIEW SIGNAL
Search and retrieval	20%	Qdrant depth, hybrid retrieval, relevance tuning
Evaluation engineering	15%	Golden sets, metrics, regression and error analysis
Enterprise integrations	15%	Auth, webhooks, events, idempotency and failure modes
Platform reliability	15%	Kubernetes, observability, SLOs and incidents
System design and FDE	15%	Discovery, architecture, rollout and customer value
Coding and SQL	10%	DSA, async Python, API exercises and SQL
Leadership and writing	10%	Staff-level influence and remote communication

Three preparation rules

1. Learn through proof

- Every topic must produce a diagram, benchmark, code sample, test or project story.
- Practise explaining the trade-off, not only defining the technology.
- Keep measurable outcomes: quality, latency, cost, uptime or customer impact.

2. Prepare at senior scope

- Start from ambiguous requirements and define the problem yourself.
- Include security, failure handling, rollout and operations in every design.
- Explain how you influence teammates, stakeholders and customers.

Readiness test: for every major topic, you should be able to define it, design it, implement a small version, debug a failure and explain one real production trade-off.

02

Retrieval, vector search and production RAG

Highest-value preparation area for Qdrant, Weaviate, Elastic and Sourcegraph.

PRIORITY 0 - MUST MASTER

Information retrieval fundamentals

- Dense, sparse and hybrid retrieval
- BM25, embeddings and semantic search
- Similarity metrics and vector normalization
- ANN and HNSW concepts
- Chunking and document structure
- Metadata filtering and payload indexes
- Query rewriting, expansion and decomposition
- Reranking and reciprocal-rank fusion
- Context packing, token budgets and citations

Proof item: Explain a hybrid-search pipeline and why each stage exists.

Relevance and performance

- Precision@k, recall@k, MRR and nDCG
- Offline relevance datasets and judgments
- Latency-versus-recall tuning
- Embedding and reranker selection
- Multi-vector and multimodal retrieval
- Caching and incremental indexing
- Filtered-vector-search performance
- Load tests and bottleneck analysis
- Search quality monitoring in production

Proof item: Benchmark three retrieval variants against one golden dataset.

Qdrant deep dive

- Collections, points, vectors and payloads
- HNSW parameters and payload indexing
- Quantization and memory trade-offs
- Sharding, replication and consistency
- Snapshots, backups and migrations
- Multi-tenant collection patterns
- Kubernetes and cloud deployment
- Performance and reliability troubleshooting

Proof item: Deploy Qdrant, ingest a document corpus, tune recall and publish the results.

Adjacent search systems

- Elasticsearch and OpenSearch architecture
- Lucene segments, inverted indexes and scoring
- Solr concepts and migration considerations
- PostgreSQL pgvector and hybrid patterns
- Vector database selection criteria
- Managed versus self-hosted trade-offs
- Data migration, reindexing and zero-downtime rollout

Proof item: Create a one-page decision matrix: Qdrant vs OpenSearch vs pgvector.

Interview target: confidently design a search system, tune its relevance, investigate slow queries and explain how you would migrate a customer's existing search workload.

03

Agentic systems and LLM application engineering

Move from framework familiarity to reliable, observable and cost-bounded production systems.

PRIORITY 0 - MUST MASTER

Agent architecture

- Tool and function calling
- Structured outputs and schema validation
- Router, planner-executor and supervisor patterns
- State machines and workflow orchestration
- Short-term and long-term memory
- Multi-agent coordination
- Human approval and escalation
- Checkpointing and resumable execution
- Termination rules and runaway-loop prevention

Proof item: Draw an agent state machine with retries, approvals and terminal states.

Models, prompts and context

- OpenAI, Anthropic and Gemini APIs
- Model selection, routing and fallback
- Prompt and tool design
- Prompt and model versioning
- Context-window management
- Streaming and partial failures
- Caching and semantic caching
- Fine-tuning and distillation decision criteria
- Cost and latency budgets

Proof item: Defend why each step is agentic, deterministic or human-controlled.

Reliability controls

- Timeouts, retries and exponential backoff
- Fallback tools and models
- Output validation and repair
- Tool permission boundaries
- MCP and connector architecture
- Observability across every agent step
- Token, latency and tool-call limits
- Safe degradation and user-visible recovery

Proof item: Implement one LangGraph workflow with traceable failures and recovery.

Senior-level judgment

- Where agents create real value
- Where deterministic code is safer
- When human judgment is mandatory
- Smallest correct system versus over-engineering
- Quality, speed and cost trade-offs
- Product impact and measurable outcomes
- Customer feedback into technical requirements

Proof item: Prepare a two-minute opinion on the limits of autonomous agents.

Sourcegraph's application directly asks for your opinion on coding agents and for a shipped multi-step system with production controls. Prepare specific, measurable examples.

04

Evaluation and AI quality engineering

The clearest gap between a demo builder and a senior production AI engineer.

PRIORITY 0 - MUST MASTER

Evaluation foundations

- Golden and representative datasets
- Baselines and acceptance thresholds
- Retrieval versus generation evaluation
- Error taxonomies and slice analysis
- Faithfulness, groundedness and relevance
- Tool selection and task completion
- Deterministic checks versus probabilistic scoring
- Dataset curation, versioning and leakage prevention

Proof item: Create a versioned golden set for one production RAG use case.

Regression and experimentation

- Unit, smoke and behavioral regression tests
- Prompt and model change gates
- LLM-as-judge design and calibration
- Human annotation rubrics
- Offline versus online evaluation
- A/B tests and production feedback
- Cost, token and latency regression
- Dashboards, alerting and release decisions

Proof item: Build a CI test that blocks a prompt release when quality drops.

Safety and privacy evaluation

- Prompt-injection test suites
- Sensitive-data and PII leakage
- Unsafe tool-use scenarios
- Cross-tenant data isolation tests
- Policy and refusal consistency
- Adversarial and malformed input
- Human escalation and incident criteria

Proof item: Add adversarial cases to the same golden dataset.

Tools to understand

- LangSmith and Langfuse
- RAGAS and DeepEval
- Arize or equivalent observability platforms
- OpenTelemetry for application traces
- Custom Python evaluation harnesses
- Experiment metadata and reproducibility
- Production sampling and feedback capture

Proof item: Know one tool deeply and compare it with two alternatives.

Whiteboard test: design an evaluation system that tells whether a retrieval, prompt or model change genuinely improved the product without hiding regressions in important customer segments.

05

Enterprise integrations and backend engineering

Essential for Forward Deployed, AI Platform and customer architecture roles.

PRIORITY 0 - MUST MASTER

Enterprise integration patterns

- REST and GraphQL APIs
- Webhooks and signature verification
- OAuth 2.0, OIDC and service accounts
- Idempotency and duplicate prevention
- Retries, jitter, circuit breakers and rate limits
- Pagination and API versioning
- Backfills, replay and reconciliation
- Data synchronization and failure recovery
- Long-running jobs and workflow status

Proof item: Design a resilient HRIS or CRM connector with replay and audit logs.

Event-driven and distributed systems

- Kafka, SQS or Pub/Sub
- At-most-once and at-least-once delivery
- Dead-letter queues
- Transactional outbox pattern
- Ordering, deduplication and backpressure
- Event schemas and evolution
- Distributed transactions and sagas
- Multi-tenant SaaS and tenant isolation
- RBAC and privacy-by-design

Proof item: Explain exactly how duplicate events are prevented or tolerated.

Python backend depth

- Advanced Python and typing
- AsyncIO, concurrency and cancellation
- FastAPI and Pydantic
- Dependency injection and background jobs
- Error handling and API contracts
- Pytest, mocking and fixtures
- Integration and contract testing
- Profiling, memory and dependency management

Proof item: Build and test one async API with a queue and idempotent worker.

Role-specific stack breadth

- Node.js and TypeScript for Deel
- Go fundamentals for Sourcegraph and infrastructure roles
- React basics for AI-powered interfaces
- GraphQL and multi-service environments
- PHP only when actively targeting Automattic
- Ability to review AI-assisted code critically

Proof item: Be production-strong in Python; gain reading and implementation fluency in one secondary stack.

06

Data systems, cloud and platform engineering

Prepare the complete path from messy enterprise data to an operable AI service.

PRIORITY 1 - STRONG WORKING DEPTH

PostgreSQL and data systems

- Schema design and normalization
- Indexes, EXPLAIN and query plans
- Transactions, isolation, locks and deadlocks
- Connection pooling and partitioning
- JSONB, full-text search and row-level security
- pgvector and hybrid search
- Migrations, backups and recovery
- Data lineage and auditability

Proof item: Optimize a slow query and explain the plan before and after.

Data ingestion and document processing

- ETL versus ELT
- Batch versus streaming ingestion
- Deduplication and data validation
- PDF, CSV and Excel parsing
- OCR and document pipelines
- Malformed and unstructured data
- Incremental processing and checkpoints
- Reprocessing, lineage and audit trails

Proof item: Show one reliable ingestion pipeline with replay and data-quality checks.

Containers and Kubernetes

- Docker images, layers and multi-stage builds
- Pods, deployments, services and ingress
- ConfigMaps, secrets and workload identity
- Readiness, liveness and startup probes
- Resource requests, limits and autoscaling
- Rolling, blue-green and canary deployments
- Helm and GitOps fundamentals

Proof item: Deploy your RAG service and Qdrant with probes, limits and rollback.

Cloud, Linux and delivery

- Linux operations and troubleshooting
- DNS, TCP/IP, TLS, proxies and load balancers
- AWS/GCP IAM, networking, storage and queues
- Managed Kubernetes and serverless services
- CI/CD and Terraform
- Public versus private cloud
- Cost estimation and capacity planning

Proof item: Create a reference architecture and monthly cost estimate.

07

Reliability, observability and security

Senior candidates are expected to own the system after deployment.

PRIORITY 1 - STRONG WORKING DEPTH

Observability and operations

- Structured logs, metrics and traces
- OpenTelemetry, Prometheus and Grafana
- Distributed tracing
- SLIs, SLOs and error budgets
- Dashboards and actionable alerts
- Load and stress testing
- Profiling and capacity planning
- On-call, incident response and postmortems
- LLM tokens, latency, quality and tool-call telemetry

Proof item: Prepare one incident story: detection, mitigation, root cause, fix and outcome.

Resilience patterns

- Timeouts and retry budgets
- Backpressure and queue protection
- Graceful degradation
- Dependency isolation and bulkheads
- Rate limits, quotas and load shedding
- Failover, recovery and disaster planning
- Mean time to detect and recover
- Safe model or provider fallback

Proof item: Run a failure drill by disabling one dependency and observing recovery.

AI application security

- Direct and indirect prompt injection
- Tool abuse and excessive agency
- Retrieval poisoning and data exfiltration
- Sensitive-data leakage
- Input/output validation
- Tool permissions and sandboxing
- Model/provider data-processing risks

Proof item: Threat-model one agent flow and define preventive plus detective controls.

Enterprise security and privacy

- OAuth/OIDC, IAM and RBAC
- Secrets and key rotation
- Encryption in transit and at rest
- PII handling, masking and retention
- Audit logging and tenant isolation
- GDPR, SOC 2 and ISO 27001 awareness
- Security review and compliance evidence

Proof item: Explain how a customer can audit access to sensitive AI inputs and outputs.

08

System design and Forward Deployed Engineering

Practise complete customer solutions, not isolated components.

PRIORITY 1 - ARCHITECTURE DEPTH

#	PRACTISE THIS DESIGN	NON-NEGOTIABLE DISCUSSION AREAS
1	Enterprise RAG platform	Ingestion, hybrid retrieval, evals, access control and rollout
2	Multi-tenant AI assistant	Tenant isolation, tools, memory, quotas, auditability
3	Large-scale vector search	Indexing, relevance, sharding, performance and migration
4	Agent platform	Approvals, checkpoints, tool security, observability and cost
5	Document AI pipeline	OCR, quality checks, replay, lineage and human review
6	LLM gateway	Routing, fallback, caching, budgets, policy and audit logs
7	Enterprise connector	OAuth, webhooks, events, reconciliation and error recovery
8	Evaluation platform	Datasets, metrics, regressions, releases and dashboards

For every architecture

- Functional and non-functional requirements
- Scale estimates and assumptions
- APIs, data model and data flows
- Failure modes and recovery
- Security, privacy and compliance
- Observability and SLOs
- Cost and capacity
- Deployment, migration and rollback
- Trade-offs and rejected alternatives

FDE customer skills

- Technical discovery and clarifying questions
- Definition of done and success metrics
- Architecture reviews and workshops
- Migration assessments
- Handling objections and unsafe requirements
- Executive communication
- Rollout and change management
- Customer value and ROI
- Reference architectures and playbooks

Interview habit: begin with questions and assumptions. A senior architect should not start drawing components before understanding users, scale, data sensitivity, failure tolerance and business success.

09

Leadership, coding and communication

Global remote roles test engineering judgment, ownership and written clarity.

Staff-level leadership stories

- Leading architecture without formal authority
- Setting standards and reference patterns
- Mentoring and raising team capability
- Design and code review
- Resolving technical disagreement
- Prioritizing under uncertainty
- Managing technical debt
- Build-versus-buy decisions
- Influencing product and customer outcomes

Proof item: Prepare six STAR stories with metrics and clear personal ownership.

Remote and written communication

- Concise technical proposals
- Architecture decision records
- Direct, asynchronous status updates
- Explaining trade-offs to non-engineers
- Structured design-review documents
- Customer workshop and presentation skills
- Clear code-review comments
- Documenting AI-assisted coding decisions

Proof item: Write a two-page design memo and present it in ten minutes.

Coding interview topics

- Arrays, strings, maps and sets
- Two pointers and sliding window
- Stacks, queues and linked lists
- Binary search and intervals
- Trees, BFS, DFS and graphs
- Heaps, backtracking and greedy
- Basic dynamic programming
- Time and space complexity

Proof item: LeetCode 75, then Top Interview 150; five quality problems weekly.

Practical engineering exercises

- SQL joins, aggregation and window functions
- API implementation
- Async Python
- Data transformations
- Production tests
- Debugging unfamiliar code
- Reading TypeScript or Go
- Verbal explanation while coding

Proof item: Complete timed mixed sets and explain every trade-off aloud.

10

Role-to-topic preparation map

Use this matrix to customize preparation before each application.

TARGET ROLE	TOPICS TO EMPHASIZE	FIT PRIORITY
Qdrant - Forward Deployed	Vector search, Qdrant, relevance, Kubernetes, NoSQL, customer workshops	Highest
Sourcegraph - Agent Engineer	Agents, retrieval, evals, model selection, cost/latency, Go/TS, leadership	Highest
Remote - Senior FDE	APIs, events, auth, RAG, evals, enterprise security, rollout, observability	Highest
Automattic - Applied AI	User-facing AI, production LLMs, product iteration, scale, writing, TS/React/PHP	High
Deel - Senior Backend AI	Node.js, PostgreSQL, AI APIs, ETL, messy data and document parsing	High
Canonical - Cloud Architect	Linux, networking, Kubernetes, cloud, data stack, workshops, presentations	High
Supabase - Platform roles	PostgreSQL, auth, storage, realtime, APIs, RLS, open source and async work	Medium

Application evidence pack

Prepare these artifacts

- One production RAG architecture diagram
- One retrieval benchmark report
- One evaluation harness repository
- One resilient agent workflow
- One customer-style reference architecture
- One incident postmortem
- Six STAR leadership stories
- One concise technical design memo

Prepare these numbers

- Data volume and user scale
- Accuracy or relevance improvement
- Latency before and after
- Cost reduction
- Reliability or failure-rate improvement
- Delivery time
- Team or stakeholder scope
- Business or customer impact

Important: Deel currently asks for 8+ years while you have about 6 years. Apply only with an honest profile and compensate through strong production scope, leadership evidence and measurable impact.

11

A practical 12-week execution sequence

Two focused hours per day, with weekly proof rather than passive reading.

WEEK	FOCUS	CORE WORK	WEEKLY PROOF
1-2	Retrieval foundations	Hybrid search, HNSW, metrics, chunking and reranking	Working benchmark
3	Qdrant depth	Indexes, filters, quantization, sharding, backups and tuning	Deployed Qdrant lab
4	Agent reliability	State, tools, validation, retries, approvals and budgets	Resilient LangGraph flow
5	Evaluation	Golden sets, error taxonomy, regression and judge calibration	CI evaluation harness
6	Integrations	OAuth, webhooks, idempotency, queues and reconciliation	Enterprise connector design
7	Data and PostgreSQL	Indexes, plans, transactions, RLS, pgvector and ETL	Query and pipeline demo
8	Kubernetes and cloud	Deployment, probes, autoscaling, IAM, Terraform and cost	Reference deployment
9	Reliability and security	OTel, SLOs, incidents, prompt injection and tenant isolation	Threat model and drill
10	System design	Four timed architectures with trade-offs and rollout	Recorded mock designs
11	Coding and SQL	Timed mixed DSA, async API and SQL exercises	Scorecard and weak areas
12	Interview simulation	Resume deep dive, STAR, FDE case and written application	Complete interview pack

What not to over-prepare

Low-return topics for these targets

- Complete foundation-model pretraining pipelines
- Optimizer mathematics and derivations
- Distributed training internals
- Mixture-of-experts routing mathematics
- State-space-model derivations
- Transformer-history trivia

Learn only when a role explicitly asks

- Deep PyTorch kernel optimization
- CUDA programming
- Research-paper reproduction
- Large-scale model training infrastructure
- Advanced fine-tuning mathematics
- Frontend specialization

Final readiness signal: you can show one credible production story for search, agents, evaluation, integration, reliability, security, customer architecture and technical leadership.

12

Official job-description references

Selected live sources reviewed for recurring requirements and interview signals.

1. Qdrant - Forward Deployed Engineer, India
https://jobs.ashbyhq.com/qdrant.tech/5b2c7960-c83d-4bc7-8cee-3843affc21a0

2. Automattic - Applied AI Engineer
https://automattic.com/work-with-us/job/applied-ai-engineer/

3. Deel - Senior Backend Engineer, AI Focus
https://jobs.ashbyhq.com/deel/2f13d55a-318b-45fa-be3e-95c8d4c8b413

4. Remote - Senior Forward Deployed Engineer
https://remote.com/openings/7774935003

5. Sourcegraph - Agent Engineer IC4
https://job-boards.greenhouse.io/sourcegraph91/jobs/6103567004

6. Canonical - Cloud Solutions Architect, Alliances
https://canonical.com/careers/5915936/cloud-solutions-architect-alliances-remote

7. Supabase - Careers
https://supabase.com/careers

Job descriptions change. Before applying, re-open the official role, confirm location eligibility and rewrite the emphasis of your resume and preparation notes for that specific opening.

Document note

This handbook is a personalized preparation guide, not a guarantee of selection. Salary, hiring stages, location eligibility and role requirements can change. Use current official career pages as the final source of truth.